Modeling Machine Learning: A Cognitive Economic Approach

Andrew Caplin[1], Daniel Martin[2], and Philip Marx[3]

Supported by Sloan Foundation Grant "Cognitive Economics at Work"

[1]New York University
[2]University of California, Santa Barbara
[3]Louisiana State University

Brown University
November 2024

Today: A One Slide Summary

- Long history in economics and cognitive science of asking whether **human behavior** is consistent with models of human cognition
- Our question: Are **machine learning (ML) predictions** consistent with standard models of human cognition?
- Our method: Treat machines exactly as we do humans $\rightarrow$ use theory to rationalize the actions they take in experiments
- Our experiment: Vary the loss function used to train CheXNeXt
  - An influential deep learning convolutional neural net for predicting diseases from chest X-ray images (Rajpurkar, Irvin, Ball, et al. 2018)
- We find that it adjusts its learning as if it was a rational human being who found learning costly, as in rational inattention models

1. Models of Human Cognition

Preliminaries

- Finite set of **outcomes** Y
  - In ML, this is the image type, sentiment, etc.
  - In decision theory and information economics, these are the states
- Set of possible **predictions** A
  - In ML, these are the confidence scores, outcomes, etc.
  - In decision theory and information economics, these are the actions
- An important input is a **loss function** $L : A \times Y \to \mathbb{R}$
  - In ML, this is cross-entropy, MSE, MAE, etc.
  - In decision theory and information economics, this is utility
  - Important: this the machine's utility, not the utility of the human using the machine (that's a separate question related to *alignment*)

Signal-Based Learning

- How is human cognition typically modeled?
- Start out with prior $\mu \in \Delta(Y)$, get a signal, end up with posterior $\gamma \in \Delta(Y)$
- $Q \in \Delta(\Delta(Y))$ is the distribution of posteriors
- $q(a|\gamma)$ is the probability that a is chosen for posterior γ
- The agent has a "strategy" (Q, q)

First Model of Learning

- How does the agent decide which strategy (Q, q) to choose?
- The first model we test is **capacity-constrained learning**
  - Characterized by Caplin, Martin, Marx, Morozova, and Xu (2024)
  - Generalizes the fixed-capacity versions of rational inattention (Sims 2003) and efficient coding (Woodford 2014)
- The agent chooses a optimal q given Q (optimal predictions given what was learned) and an optimal Q from a fixed feasible set of Q's (optimal learning given what can be learned)

Formal statement

Interpretation for Machine Learning

- Capacity-constrained learning imagines an algorithm as having a fixed set of mathematical operations it can use
- And as selecting among these mathematical operations to best match the incentives provided by the loss function
- Is this the "folk theory" of ML?

"Making the best predictions that it can"

Second Model of Learning

- What is another way that the agent could decide which (Q, q) to choose?
- A second possibility, **costly learning**, assumes the algorithm chooses from Q's of varying costs that are additively separable
  - Characterized by Caplin and Dean (2015)
  - Generalizes the specialized version using Shannon entropy characterized by Matejka and McKay (2015)
  - Nest capacity-constrained learning as a special case

Formal statement

Interpretation for Machine Learning

- Costly learning imagines that there is relative costs in performing different mathematical operations, and the algorithm chooses them based on the incentives provided by the loss function
- Agnostic on how it implements cost/benefit tradeoff in practice
- These costs are the algorithm's "pseudo" costs: how costly it treats performing different mathematical operations
  - Food for thought: we might care if the algorithmic pseudo-costs reflect the "real" costs running the algorithm (run-time, power, etc.)
- Connection to CS literature on *implicit regularization*: implicit regularization term analogous to implicit pseudo-cost

2. Testable Conditions

Starting Point

- $P^L : A \times Y \to [0, 1]$ is the joint distribution of predictions and outcomes on a data set when training with loss function L
  - This *confusion matrix* is used to judge machine performance
  - Also, an example of "state-dependent stochastic choice data" (SDSC) useful in modeling perception (Caplin and Martin 2015)
- Decision theoretic perspective: goal is to understand mapping from $L \in \mathcal{L}$ to performance data P^L
- If this mapping satisfies certain counterfactuals, then it is consistent with capacity-constrained learning and/or costly learning

Condition 1: Loss-Calibrated

- An algorithm is **loss-calibrated** to loss function L if a wholesale switch of predictions does not reduce losses according to L:

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

  - Restatement of the No Improving Action Switches (NIAS) condition of Caplin and Martin (2015) in the machine learning setting
- Normatively-grounded: rules out trivial prediction errors
- Intuitive: algorithm reporting correctly given it's loss function
- Counterfactual: imagine switching prediction a at a posterior belief where chose a to feasible unchosen alternative a'

Value of Learning Matrix G

- Because indexing will be useful, in what follows we will take as given a finite set of M loss functions, indexed by $1 \leq m \leq M$
- Key is the *value of learning matrix* G with generic element G^{mn}:

$$G^{mn} \equiv \sum_{a \in A} \min_{a' \in A} \sum_{y \in Y} L^m(a', y) P^n(a, y)$$

- Gives minimized expected losses when loss function is L^m and loss function L^n is used for training
- RHS wholesale replaces any prediction a with alternative prediction a', computes the expected losses for L^m, and then minimizes

Illustration

Condition 2: Strongly Loss Adapted

- Counterfactual: cannot be possible for any loss function to lower losses with learning revealed to be feasible for different loss function
- An algorithm is **strongly loss-adapted** if for all $1 \leq m, n \leq M$,

$$G^{mn} \geq G^{mm}$$

  - Restatement of the No Improving Switches (NIS) condition of Caplin, Martin, Marx, Morozova, and Xu (2024) in machine learning setting
- Together with loss-calibration, strongly loss-adapted is necessary and sufficient for capacity-constrained learning

Condition 3: Strongly Loss Adapted

- Characterizing costly learning is based on paths of switches
- Readily visible in the **indirect value difference matrix** D, which has generic element D^{mn} defining the minimum change in the sum of losses by switching loss functions along a path from m to n
- Formally, an algorithm with is **loss-adapted** if for all $1 \leq m \leq M$,

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

- This condition requires that no cycle of switches could lower losses
  - Restatement of the No Improving Attention Cycles (NIAC) condition of Caplin and Dean (2015) in the machine learning setting
- An algorithm has a costly learning explanation if and only if it is loss-calibrated and loss-adapted

Indirect Value Difference Matrix D

- Counterfactual: rule out loss reduction in cost-preserving cycles of learning
- Tightly analogous to SARP for budget-constrained choices
- If an algorithm is loss adapted, then the indirect value difference matrix D can be computed by applying the Floyd-Warshall algorithm to the complete weighted directed graph with weight $D_0^{mn} = G^{mn} - G^{mm}$ on the directed edge from node m to node n
- An algorithm is loss adapted if and only if the resulting D matrix thus computed has a zero diagonal

Illustration

3. Testing

Algorithm

- We consider a deep learning neural net used to predict pneumonia in the "ChestX-ray14" dataset (Wang et al. 2017)
- The ChextX-ray14 dataset consists of 112,120 frontal chest X-rays which were labeled with thoracic diseases
- Essentially a replication from the publicly available codebase of Rajpurkar et al. (2018)
- In the binary classification task, the outcomes of interest are pneumonia ($y = 1$) or not ($y = 0$)
- For each X-ray, the trained model reports a confidence score $a \in [0, 1]$

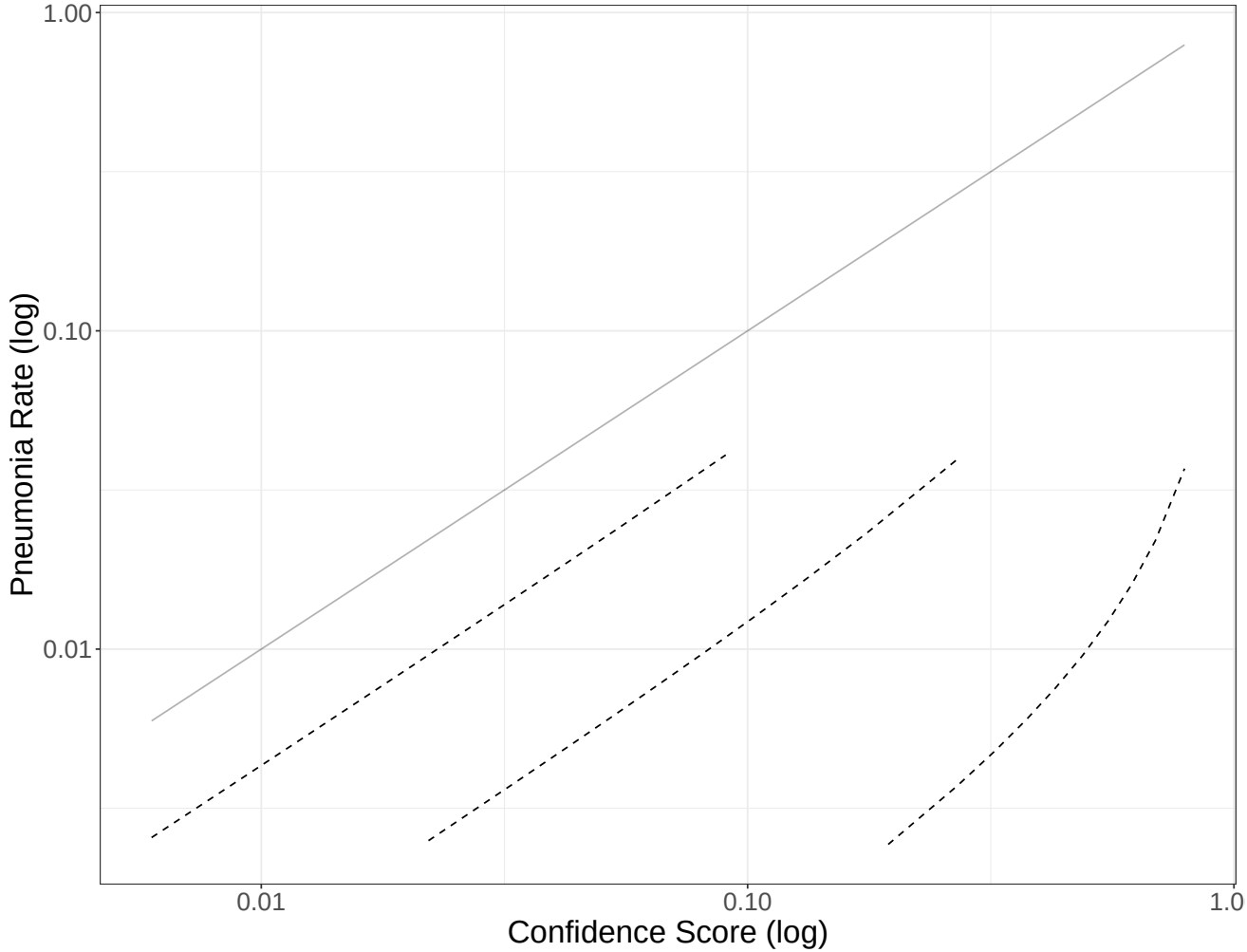
Experiment

- For this task, the standard loss function is the literature is weighted cross-entropy

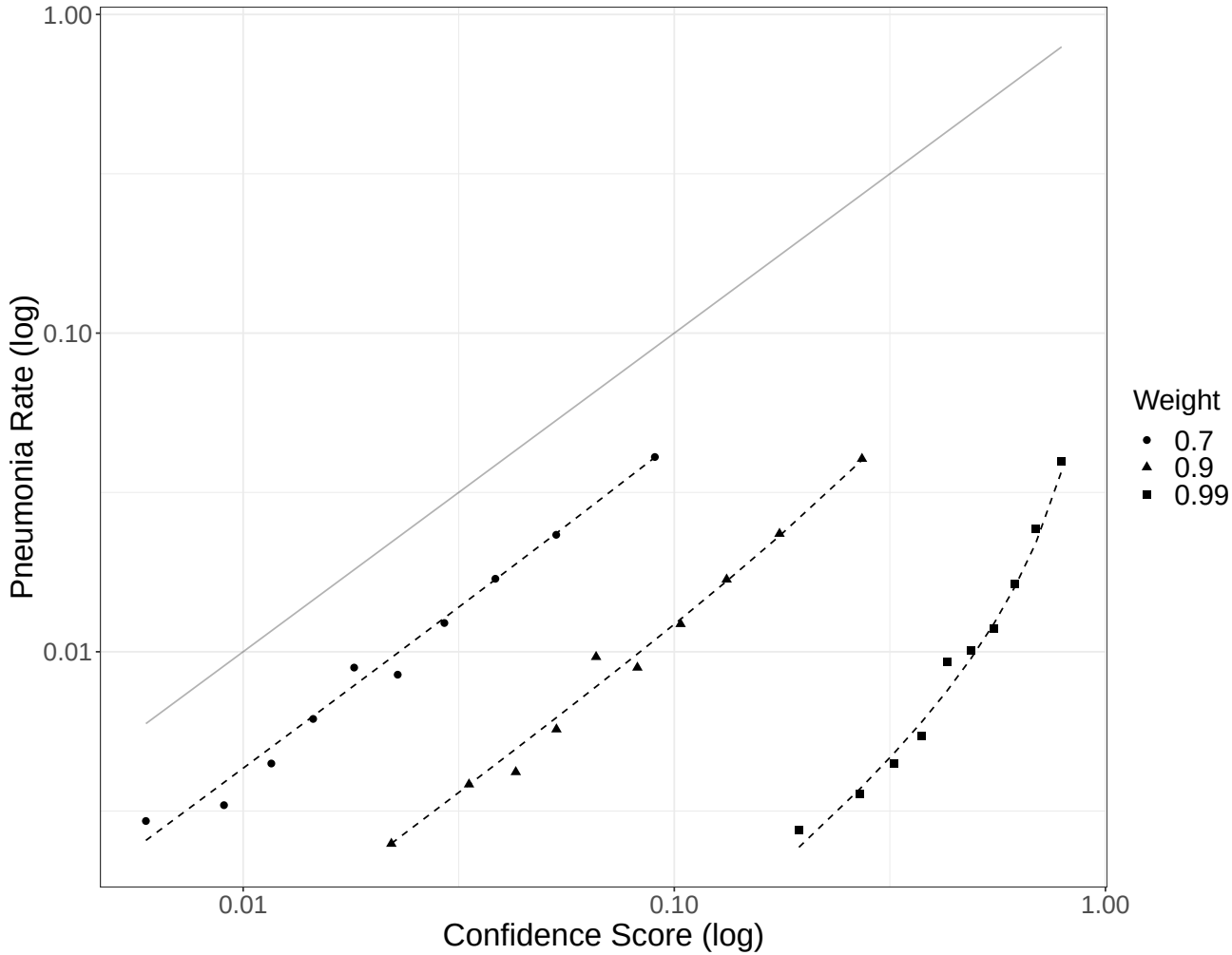
$$L(a, y) = \begin{cases} -\beta log(a) \text{ if } y = 1 \\ -(1-\beta) log(1-a) \text{ if } y = 0 \end{cases}$$

- But pneumonia is underrepresented, so natural to add class weights with values of the weight parameter $\beta > 0.5$ indicating a greater penalty for false negative than false positive prediction of pneumonia ($\beta \approx 0.99$ in Rajpurkar et al. 2018)
- Our experiment: Vary the loss function with class weights $\beta = 0.7, 0.9, 0.99$
- First question: Is the algorithm loss-calibrated?

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: It becomes increasingly miscalibrated with higher weights, but is loss-calibrated



Capacity-Constrained Learning

- To test the other conditions, we generate the G matrix

$$G = 0.01 \begin{array}{c} \begin{matrix} P^{0.7} & P^{0.9} & P^{0.99} \end{matrix} \\ \begin{pmatrix} 3.750 & 3.752 & 3.762 \\ 3.349 & 3.352 & 3.362 \\ 1.365 & 1.366 & 1.370 \end{pmatrix} \end{array} \begin{matrix} L^{0.7} \\ L^{0.9} \\ L^{0.99} \end{matrix}$$

- First we look at whether the algorithm is strongly loss-adapted, and thereby consistent with capacity-constrained learning:

$$\mathcal{H}_0 : G^{mn} \geq G^{mm} \quad \text{for all } 1 \leq m, n \leq M$$

- Performed statistical test by bootstrapping covariance matrix
  - Bonferroni correction on set of one-sided Wald tests, $p = 0.0014$
  - Conservative test for multivariate one-sided, yet still reject

Costly Learning

- Next we look at whether the algorithm is loss-adapted, and thereby consistent with costly learning:

$$\mathcal{H}_0 : D^{mm} \geq 0 \quad \text{for all } 1 \leq m \leq M$$

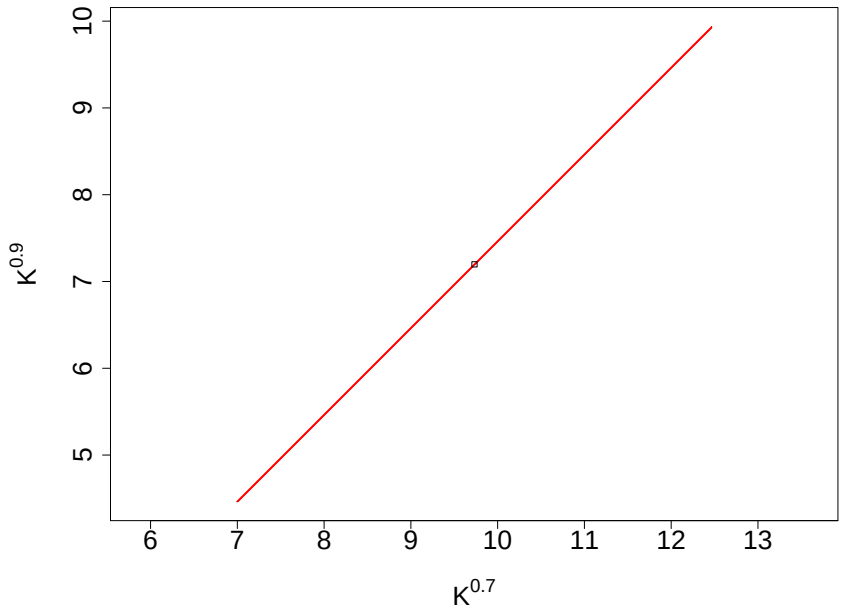
- The estimated D matrix, given below, satisfies this null hypothesis

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

- We therefore fail to reject that the algorithm is consistent with costly learning at any level of significance

Results

- Algorithm is loss-adapted, so can recover costs



- Normalize to $K^{0.99} = 0$
- The relationship between $K^{0.7}$ and $K^{0.9}$ is nearly point identified
- Interpretation: 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

Open Questions

- What are the implications of this model for machine learning?
- Are machine learners consistent with specialized models of rational inattention (with particular functional forms of cost)?
- Are there non-rationalizable cases (e.g., Frank, Gao, and Yang 2024) and can existing behavioral models cover these cases?

What Form of Economic Theory Is This?

- Stigler (1976) claimed that path forward in understanding productive efficiency requires better modeling of mistakes:

  Waste is error within the framework of modern economic analysis, and it will not become a useful concept until we have a theory of error.

- This form of theory increasingly in place: rational inattention, efficient coding, probabilistic distortions
- Data innovations also liberating progress: SDSC data central in this emerging efforts

What Form of Economic Theory Is This?

- Patterns in mistakes central in the age of AI
- Theory will play a far bigger role in the age of AI than in the agricultural, manufacturing, and service economies given ability to study mistakes in many AI applications

Future Directions

1. Can use our method/models to determine optimal loss function given your objective ("loss function design") – tie to **alignment** literature
2. Can assess whether the algorithm's pseudo-costs reflect the real costs of running the algorithm
3. Can use to categorize algorithms more generally (including non-classification algorithms and simple techniques like MLE)
4. Can use to understand impact of different programming approaches
5. Can use to re-calibrate algorithms (recover "beliefs")
6. And many, many others!

Thank You!

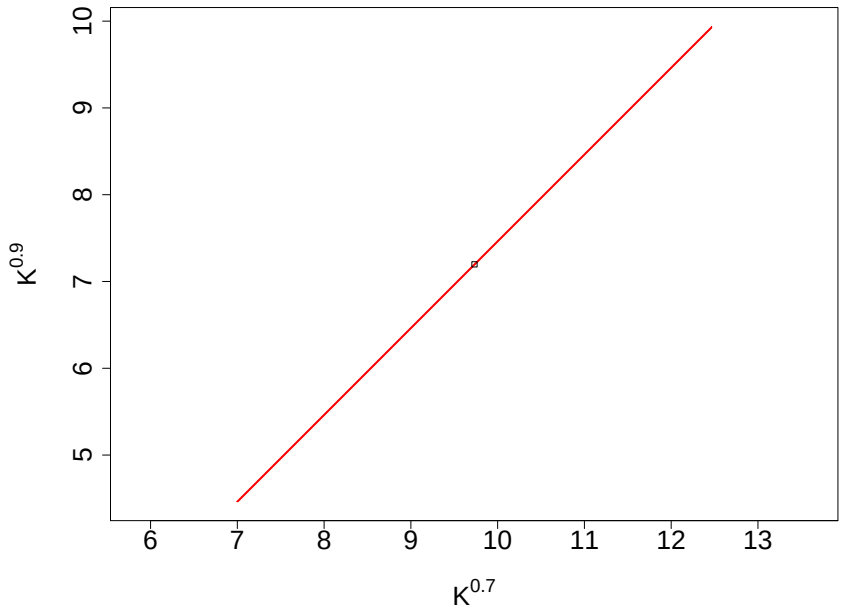
daniel@martinonline.org

- Building on Caplin and Dean (2015), Caplin, Martin, and Marx (2023) show how to identify all costs $\{K^m\}_{m=1}^M$ that rationalize the observed performance data according to costly learning
- For present purposes, a key observation is that, normalizing to $K^M = 0$, learning costs $\{K^m\}_{m=1}^M$ define a convex polyhedron in $\mathbb{R}^{M-1}$ with the sign-inverted M-th row $(-D^{M1}, \cdots, -D^{M(M-1)})$ and M-th column $(D^{1M}, \cdots, D^{(M-1)M})$ providing a subset of the extreme points
- Furthermore, the average cost $\bar{K}$ across normalizations is potentially appealing as a representative learning cost because it is "central" and easy to compute

Illustration

Results

- Can recover set of possible learning costs



- The relationship between $K^{0.7}$ and $K^{0.9}$ is nearly point identified
- For any rationalizing cost function, $K^{0.99} \leq K^{0.9} \leq K^{0.7}$
- Interpretation: 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

Illustration of G

- There are three steps to calculate G^{mn}
  1. Determine the theoretically-implied posterior "beliefs" for each action in P^n under loss function L^n
  2. Determine the theoretically-optimal confidence score to report under loss function L^m for each implied belief
  3. Calculate losses under L^m for the theoretically-optimal confidence score under L^m at corresponding probabilities in P^n
- Will take these steps in the following simple stylized example that imagines just two confidence scores are reported
- Will also take these steps in the empirical implementation, in which a great many confidence scores are reported

Back

Recall cross-entropy: $L(a, y) = -\beta y \log a - (1 - \beta)(1 - y) \log 1 - a$

Imagine the algorithm produces this performance data for $\beta = 0.5, 0.7$:

$$P^1 = \begin{array}{c} \begin{matrix} y=0 & y=1 \end{matrix} \\ \begin{pmatrix} .3 & .2 \\ .2 & .3 \end{pmatrix} \end{array} \begin{array}{l} a = .4 \\ a = .6 \end{array} \quad \& \quad P^2 = \begin{array}{c} \begin{matrix} y=0 & y=1 \end{matrix} \\ \begin{pmatrix} .3 & .1 \\ .2 & .4 \end{pmatrix} \end{array} \begin{array}{l} a = .44 \\ a = .82 \end{array}$$

G^{11}: For P^1 the implied beliefs under L^1 are .4 and .6 (because proper), which are also optimal reports under L^1 (because proper), so losses of .34

G^{12}: For P^2 the reported confidence scores are .44 and .82, but the implied beliefs under L^2 are .25 and .66, which are optimal reports under L^1 (because it is proper), so losses of .30

Repeating this, the value of learning matrix can be computed as:

$$G = \begin{array}{c} \begin{matrix} P^1 & P^2 \end{matrix} \\ \begin{pmatrix} .34 & .30 \\ .30 & .27 \end{pmatrix} \end{array} \begin{array}{l} L^1 \\ L^2 \end{array}$$

To illustrate, consider again the following G matrix:

$$G = \begin{array}{c} \\ L^1 \\ L^2 \end{array} \begin{array}{c} \begin{array}{cc} P^1 & P^2 \end{array} \\ \begin{pmatrix} .34 & .30 \\ .30 & .27 \end{pmatrix} \end{array}$$

The corresponding value difference matrix is

$$D = \begin{array}{c} \\ L^1 \\ L^2 \end{array} \begin{array}{c} \begin{array}{cc} P^1 & P^2 \end{array} \\ \begin{pmatrix} -.01 & -.04 \\ .03 & -.01 \end{pmatrix} \end{array}$$

$D^{11} = -.01$ because starting from L^1 the indirect path back to L^1 involves first lowering losses by .04 switching P^1 to P^2 and then raising losses by .03 switching P^2 to P^1

Back

Illustration of Recovery

- Consider a loss-calibrated algorithm for which we specify the corresponding G matrix directly as:

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

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

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

Illustration of Recovery

- 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 capacity-constrained machine learning, but it is consistent with costly learning

Illustration of Recovery

- 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

Illustration of Recovery

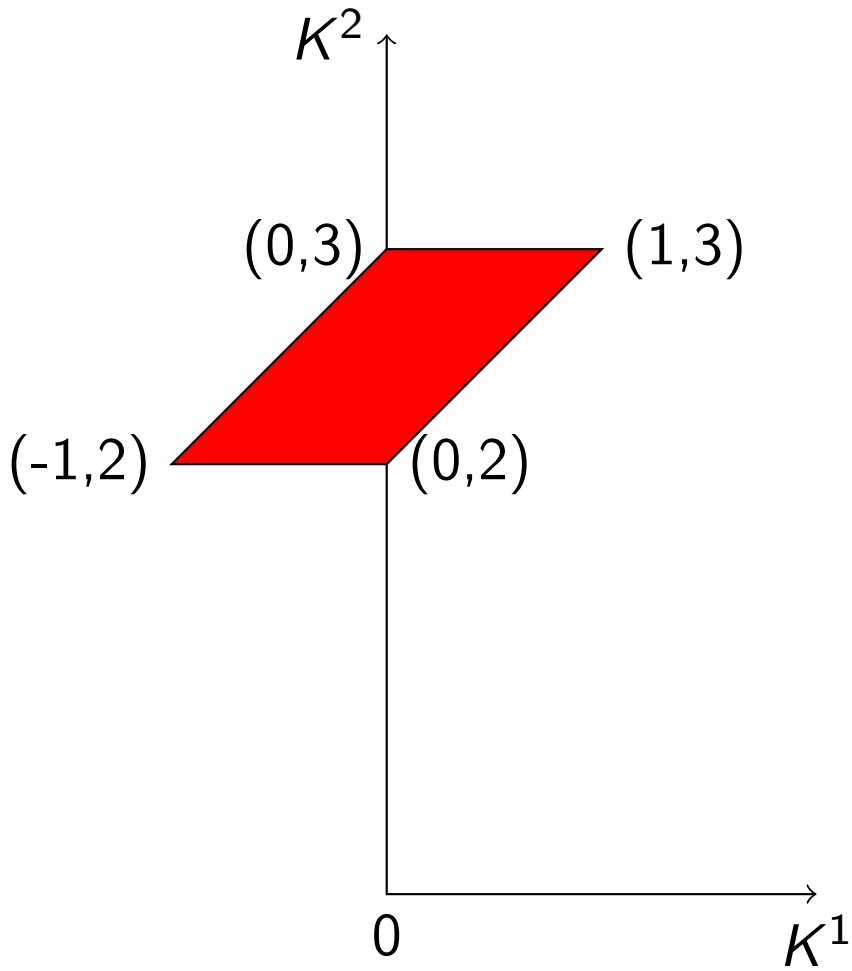


Figure 1. Example 2-dimensional learning cost polyhedron.

Illustration of 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

Back

Formal Statement (Capacity-Constrained)

- We now formally define the algorithm's strategy space Λ

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

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

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

Back

Formal Statement (Costly Learning)

- Define a cost function $K : \mathcal{Q} \to \bar{\mathbb{R}}$
- The **cost adjusted loss** $\hat{L}$ of strategy (Q, q) is

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

- (Q, q) is selected to maximize $\hat{L}(Q, q|L, K)$ given L and K

Back