Feature Selection as a Two-Player Game

Praneet Rathi & Malcolm Forsyth

CS 580

November 2023

Table of Contents

Introduction

- **General Framework**: Given a dataset and target (classification or regression), for a specific model, select a small number of features that are as robust as the full set.
- **Motivation**: 1) Feature selection is an important problem in Machine Learning, to avoid overfitting and reduce dimensionality [Liu+21]. 2) Adversarial robustness is another important problem, especially in regression problems to avoid the impact of outliers [MW05].
- **Idea**: Tackle these two problems at once by framing feature selection and adversarial robustness as a game and attempt to solve for a Nash, which ideally yields both a feature selection method and an adversarial examples method.

Individual Problem Statements

Consider an $n \times m$ labeled dataset $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$ $\mathcal{X} \in \mathbb{R}^{n \times m}, \mathcal{Y} \in \mathbb{R}^n$

From Motivation 1:

- Seek to find a small set of k features from the dataset that are nearly as performant and robust as the full set.
- The true objective is to minimize some penalty on a test set.

From Motivation 2:

- Seek to identify a set of adversarial samples in the dataset that are difficult to perform well on.
- The true objective is to maximize some penalty for a trained model.

Joint Problem Statement

Two Player Game:

- The *column-player* to be an actor which seeks to select k columns (features), S_{column}
- The *row-player* seeks to select l rows which are hard to predict, S_{row}

New data matrix is formed from the selected features and examples:

$$X = \begin{pmatrix} x_{i_1 j_1} & x_{i_1 j_2} & \cdots & x_{i_1 j_l} \\ x_{i_2 j_1} & x_{i_2 j_2} & \cdots & x_{i_2 j_l} \\ \vdots & \vdots & \ddots & \vdots \\ x_{i_k j_1} & x_{i_k j_2} & \cdots & x_{i_k j_l} \end{pmatrix} \quad i \in S_{row}, j \in S_{column}$$

Problem Statement

Each player is then scored based on how X predicts y; Ex. if we consider linear regression via least squares, we take estimator

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

And we score the players on the classic R^2 objective

$$R^2 = 1 - \frac{(y - X\hat{\beta})^T (y - X\hat{\beta})}{(y - \bar{y})^T (y - \bar{y})}$$

At the end of the game, the column player receives a payoff of R^2 and the row player receives a payoff of $-R^2$.

Problem Statement

Importantly, we are operating over the space of subsets, so our game has a normal form which is exponential in both dimensions. Only payoff structure:

$$S_i \subset S_j \implies \forall T_k : R^2_{T_k,S_i} \leq R^2_{T_k,S_j}$$

This doesn't give much ... even computing whether a given point is Nash remains exponential.

	S_1	S_2	$\cdots$	$S_{\binom{m}{k}}$
T_1	$(R^2_{T_1,S_1}, -R^2_{T_1,S_1})$	$(R^2_{T_1,S_2}, -R^2_{T_1,S_2})$	$\cdots$	$(R^2_{T_1,S_{\binom{m}{k}}}, -R^2_{T_1,S_{\binom{m}{k}}})$
T_2	$(R^2_{T_2,S_1}, -R^2_{T_2,S_1})$	$(R^2_{T_2,S_2}, -R^2_{T_2,S_2})$	$\cdots$	$(R^2_{T_2,S_{\binom{m}{k}}}, -R^2_{T_2,S_{\binom{m}{k}}})$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$T_{\binom{n}{l}}$	$(R^2_{T_{\binom{n}{l}},S_1}, -R^2_{T_{\binom{n}{l}},S_1})$	$(R^2_{T_{\binom{n}{l}},S_2}, -R^2_{T_{\binom{n}{l}},S_2})$	$\cdots$	$(R^2_{T_{\binom{n}{l}},S_{\binom{m}{k}}}, -R^2_{T_{\binom{n}{l}},S_{\binom{m}{k}}})$

State of the Art Methods: Feature Selection

- **LASSO Regression:** A simple and widely adopted technique where a linear model is taken, seeking β to minimize the penalty

$$(y - X\beta)^T(y - X\beta) + \lambda||\beta||_1$$

  parameterized by some positive lambda. Because of the 1-norm, the model tends to shrink weights to zero, and therefore you can increase λ until k features are remaining [Tib96]
- **Minimal Redundancy Maximal Relevance (mRMR):** Based on information theory, incrementally selects features with high information to the target while pruning mutually dependent features [PLD05]
- **Multi-Agent Reinforcement Learning Feature Selection (MARLFS):** RL model where each feature has a corresponding agent with the option to include or exclude the feature [Liu+21]

State of the Art Methods: Adversarial Robustness

- **Stratified Boostrap Selection:** compute residuals on data points, then stratify the data points by their residual rankings, then created bootstrapped stratified samples across the different groups of residuals [MW05]
- **Elipsoidal Peeling:** to remove outliers, take a minimal elipsoid (or other convex shape) around the data, and remove points which lie on the boundary, and repeat [LKA13]

Theory

- We have that for non-degenerate games (which may be reasonable for a data matrix), at a Nash equilibrium, the support of the row player and the column player is equal [Rou10].
  - (sketch) general position data matrix $\implies$ general position game matrix $\implies$ non-degenerate
- If we had that the unique best response for the column player to any strategy by the row player was pure, then we would have that all Nash equilibria for our game would be pure (still working on this).

Discrete Optimization

Monte Carlo Tree Search (MCTS):

- For a game with many turns, keep a game tree with each node storing the number of times its been explored and the average winrate/payout. Successfully used in Chess/Go algorithms like AlphaZero [Sil+17]
- At each iteration, start from the root and choose the child action which maximizes the *Upper Confidence Bound applied to Trees (UCT)* [KS06]

$$\frac{V}{n} + c\sqrt{\frac{\ln N}{n}}$$

- When we reach a node with no children who have been explored, randomly rollout to generate data

Discrete Optimization

- To apply MCTS to our game, we approximate the game where players take turns removing a feature/row until there are k features and l rows remaining
- Implemented to experiment, but the algorithm is slow because the discretized game tree is extremely large (inefficient due to permutations of removals being counted as distinct)
- For a 100×100 game where we seek to select 10 features and examples, the original game is $\binom{100}{10}^2 \approx 10^{27}$ but the MCTS game tree is $100^{(10)2} \approx 10^{39}$

Continuous Optimization

- The discrete space is quite large, what if we try continuously optimizing some function instead?
- What's the function? $F(p^*, q^*) = E_{p \sim p^*, q \sim q^*}[R^2(S_p, T_q)]$ for strategies p^*, q^*
- Considering 100×100 game, there are 2^{100} total subsets and even if we fix some k size to select, still $\binom{100}{k}$
- How to reduce dimension?

Continuous Optimization

- Instead of directly parametrizing subset as $p^* \subset \{0,1\}^{\binom{100}{k}}$, instead use low-dimensional $p^* \subset \{0,1\}^k$.
- $P(p|p^*) = \prod_{i=1}^{1} 00(p_i^*)^(I(p_i) = 1) * (1 - p_i^*)^(I(p_i) = 0)$.
- Note that $\exists i : p_i^* \notin \{0,1\} \implies \exists p : P(p|p^*) > 0$ and $|p| > k$

Continuous Optimization- Best Response Dynamics

- Recall $F(p^*, q^*) = E_{p \sim p^*, q \sim q^*}[R^2(S_p, T_q)]$:
- One player wants to maximize, one player wants to minimize, let's take the derivative:
- $\frac{dF(p^*,q^*)}{p^*} = \frac{dF(p^*,q^*)}{p_1^*} \cdots \frac{dF(p^*,q^*)}{p_n^*}$.
- $F(p^*, q^*) = p_1^* E_{p \sim p^* \text{st} p_1 = 1, q \sim q^*}[R^2(S_p, T_q)] + (1 - p_1^*) * E_{p \sim p^* \text{st} p_1 = 0, q \sim q^*}[R^2(S_p, T_q)]$.
- For a deterministic model and metric, the two expected values are fixed, so we can write:
- $\frac{dF(p^*,q^*)}{p_1^*} = E_{p \sim p^* \text{st} p_1 = 1, q \sim q^*}[R^2(S_p, T_q)] - E_{p \sim p^* \text{st} p_1 = 0, q \sim q^*}[R^2(S_p, T_q)]$.
- We can similarly write the derivative for all components
- They are fixed but how to estimate? We sample.

Continuous Optimization- Best Response Dynamics

- With these gradients, we can now jointly optimize our function through projected stochastic gradient descent.
- Problem: there are no guarantees for optimizing functions that are constantly changing-ie constants are practically functions of the input variables. Thus requires information about model and metric to provide any convergence properties.
- Intuition Gained: This can, however, be thought of as best response dynamics. Given opponent's current play, which lends itself to R^2 scores, what is our best response.

Continuous Optimization- Nash Equilibrium

- Sandholm et al. showed how to optimize over any two-person, zero-sum game with known matrix A [Hod+10].
- Key ingredients:
  - Construct prox function- none exists for our use
  - Knowing A- we don't

Continuous Optimization- Nash Equilibrium

- Previous prox function $d(x) = \sum_i x_i \log(x_i) + C$ exists for $x \in \{x : \sum_i x_i = 1, \forall i : 0 \leq x_i \leq 1\}$
- Our space: $x \in \{x : \sum_i x_i = k, \forall i : 0 \leq x_i \leq 1\}$
- We use same function, but find the new conjugate and conjugate gradient.

Continuous Optimization- Nash Equilibrium

- Optimization consists of calls to Ap and $A^T q$ for strategies p, q.
- Recall parametrization:
  $P(p|p^*) = \prod_{i=1}^{1} 00(p_i^*)^(I(p_i) = 1) * (1 - p_i^*)^(I(p_i) = 0)$
- Given p^*, we want to estimate $p \rightarrow$ estimate $Ap \rightarrow$ estimate Ap^*
- $p^* \rightarrow p$: Randomly sample pure strategies
- $p \rightarrow Ap$: Randomly sample opponent strategies to get EV for sampled pures
- $Ap \rightarrow Ap^*$: Project down to feature space

Continuous Optimization- Nash Equilibrium

- $Ap \rightarrow Ap^*$: Project down to feature space
- Key insight is our prox-based function uses updates based on exponential. So we can directly produce the most accurate update with respect to given samples, by fitting log linear regression.

	S_1	S_2	$\cdots$	S_m
T_1	1	1	$\cdots$	$\log(R_1^2)$
T_2	0	1	$\cdots$	$\log(R_2^2)$
T_3	1	0	$\cdots$	$\log(R_2^2)$

- Scores are not scale invariant, so can apply some monotonic $g(x)$ (ex. $g(x) = x^10$) on fitted coefficients.

Continuous Optimization- Nash Equilibrium

- Pros:
  - Theoretically should give us Nash, any losses are with $Ap \rightarrow Ap^*$ estimation
  - Makes no assumptions about model form, requires only fast queries and provided metric
  - Example use cases: linear regression, logistic regression, decision tree regression.
- Cons:
  - Requires fine-tuning and needs fast metric query ability

References I

[Hod+10] Samid Hoda et al. "Smoothing techniques for computing Nash equilibria of sequential games". In: *Mathematics of Operations Research* 35.2 (2010), pp. 494–512.

[KS06] Levente Kocsis and Csaba Szepesvári. "Bandit based monte-carlo planning". In: *European conference on machine learning*. Springer. 2006, pp. 282–293.

[Liu+21] Kunpeng Liu et al. "Automated feature selection: A reinforcement learning perspective". In: *IEEE Transactions on Knowledge and Data Engineering* (2021).

[LKA13] SI Lyashko, DA Klyushin, and VV Alexeyenko. "Multivariate ranking using elliptical peeling". In: *Cybernetics and Systems Analysis* 49 (2013), pp. 511–516.

References II

[MW05] Samuel Müller and AH Welsh. "Outlier robust model selection in linear regression". In: *Journal of the American Statistical Association* 100.472 (2005), pp. 1297–1310.

[PLD05] Hanchuan Peng, Fuhui Long, and C. Ding. "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.8 (2005), pp. 1226–1238. DOI: 10.1109/TPAMI.2005.159.

[Rou10] Tim Roughgarden. "Algorithmic game theory". In: *Communications of the ACM* 53.7 (2010), pp. 78–86.

[Sil+17] David Silver et al. "Mastering chess and shogi by self-play with a general reinforcement learning algorithm". In: *arXiv preprint arXiv:1712.01815* (2017).

References III

[Tib96] Robert Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 58.1 (1996), pp. 267–288.