Gradient Based Algorithms in Regressions

Version 1.0

Dr. Ye Luo
HKU Business School

Jan 2023

Gradient Based Algorithms

- ML problems are often formulated as a minimization/maximization problem.
- For example, in Linear Regression or Series Regression, an L-2 loss function is to be minimized:

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{n} (Y_i - f(X_i))^2.$$

- The class of functions $\mathcal{F}$ can be specified as linear functions of X, leading to linear regression. If $\mathcal{F}$ is specified as linear functions of Series, e.g., power series of x, then it leads to Series Regression.

Computational Issues for Large Scale Models

- It is well known that given predictor matrix X of size $n \times p$, and target of prediction Y of size $n \times 1$, the linear regression can be written as

$$\hat{\beta} = (X^\intercal X)^{-1} X^\intercal Y.$$

- When p is large, computing $(X^\intercal X)^{-1}$ takes time proportional to p^3, which could be slow.
- Therefore, alternative computational approaches need to be developed for computing large scale ML, even for simple linear regression.

Model Parametrization

- For general Loss function $L_n(f)$, the optimization problem can be much more difficult than linear regression - non-linearity, non-convexity, etc.
- A general approach for finding minimal value of $L_n(f)$ is through the "Gradient" approach.
- The gradient approach takes some parametrization form of f, e.g., $f(X) = g(X, \beta)$, where g is fixed, and $\beta \in \mathbb{R}^p$ is a parameter of the model. For example:
  linear model $f(X) = X^\intercal \beta$,
  logit model $f(X) = \frac{\exp(X^\intercal \beta)}{1+\exp(X^\intercal \beta)}$.
- The Loss function $L_n(f)$ can be rewritten as a function of the parameter β, e.g., $L_n(\beta)$.

The Gradient Space

- After parametrization, the problem reduces to find the β that minimizes $L_n(\beta)$. That said,
$$\beta = \mathit{argmin}_{\beta \in \mathbb{R}^p} L_n(\beta).$$
- Suppose that $L_n(\cdot)$ is a differentiable function. β being the minimizer of $L_n(\cdot)$ implies that
$$\nabla L_n(\beta) = 0,$$
where $\nabla L_n(\beta) = \frac{\partial L_n}{\partial \beta}$ is the partial derivative of L_n with respect to the vector β, or sometimes referred as the "Gradient".
- As a result, the problem reduces to solve the equation of
$$\nabla L_n(\beta) = 0,$$
in the Gradient space.

The Gradient Descend Algorithm

- The most widely used algorithm in ML is the "Gradient Descend" Algorithm, which takes the following recursive form:

$$\beta_{t+1} = \beta_t - \alpha_t \nabla L_n(\beta_t).$$

- Starting from β_0, the value β_t update itself in each period of time.
- α_t is called learning rate, as it defines how fast each time the parameter β moves. α_t governs the speed of learning.

The choice of α_t

- The larger the α_t is, the faster the learning process is, leading to smaller bias (bias shrinks faster).
- Large α_t can lead to instability of learning, resulting in large variance.
- Here again we meet bias-variance trade-off.
- Popular α schemes:
  1. Constant Scheme: $\alpha_t = \alpha$ is a constant. Typically α is set as a small number, e.g., 10^{-3}.
  2. $\alpha_t = \alpha_0 t^{-\delta}$, with $\delta \in (\frac{1}{2}, 1]$, and α_0 being a constant.

GD

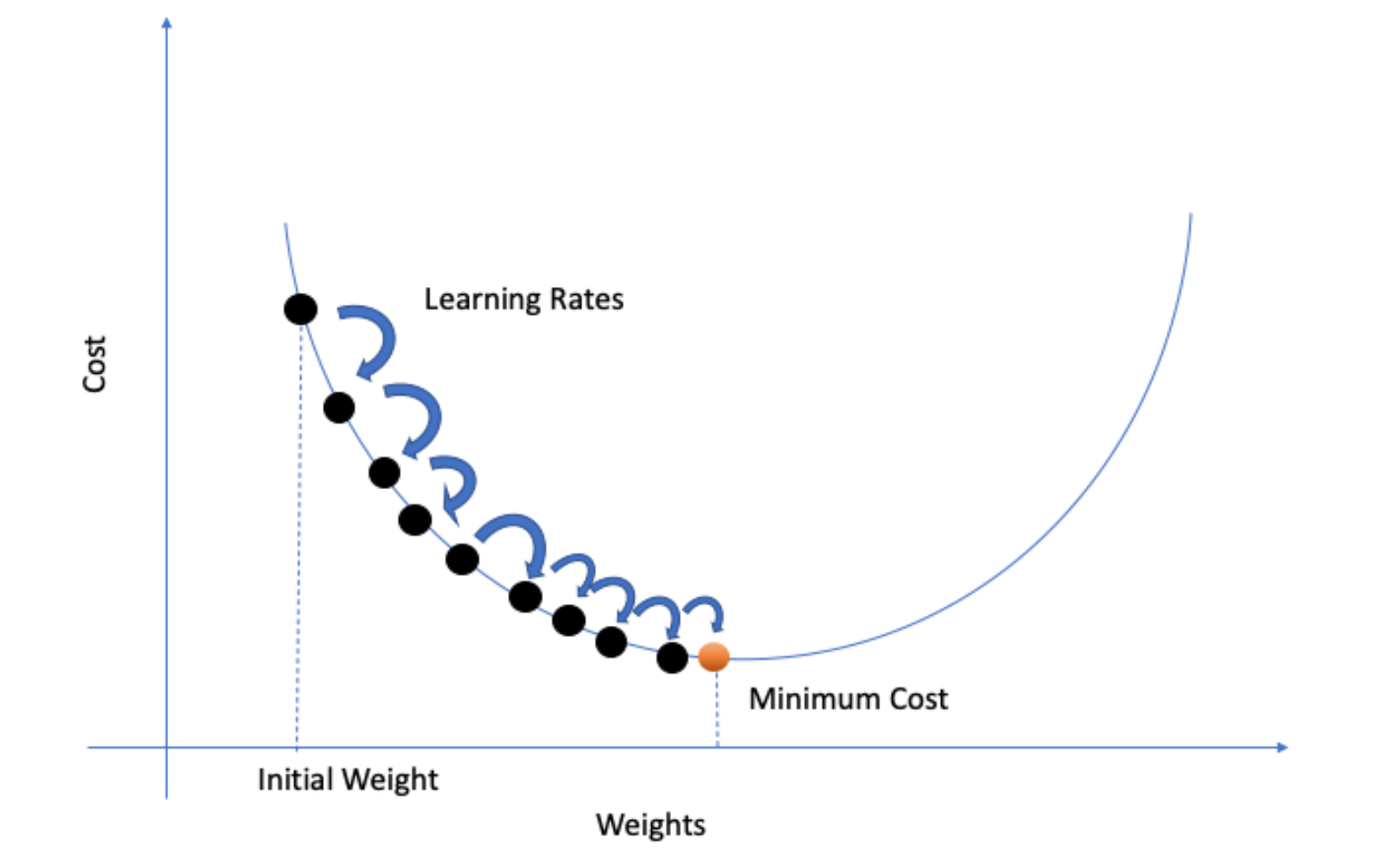

Figure 1: GD

GD in 3d

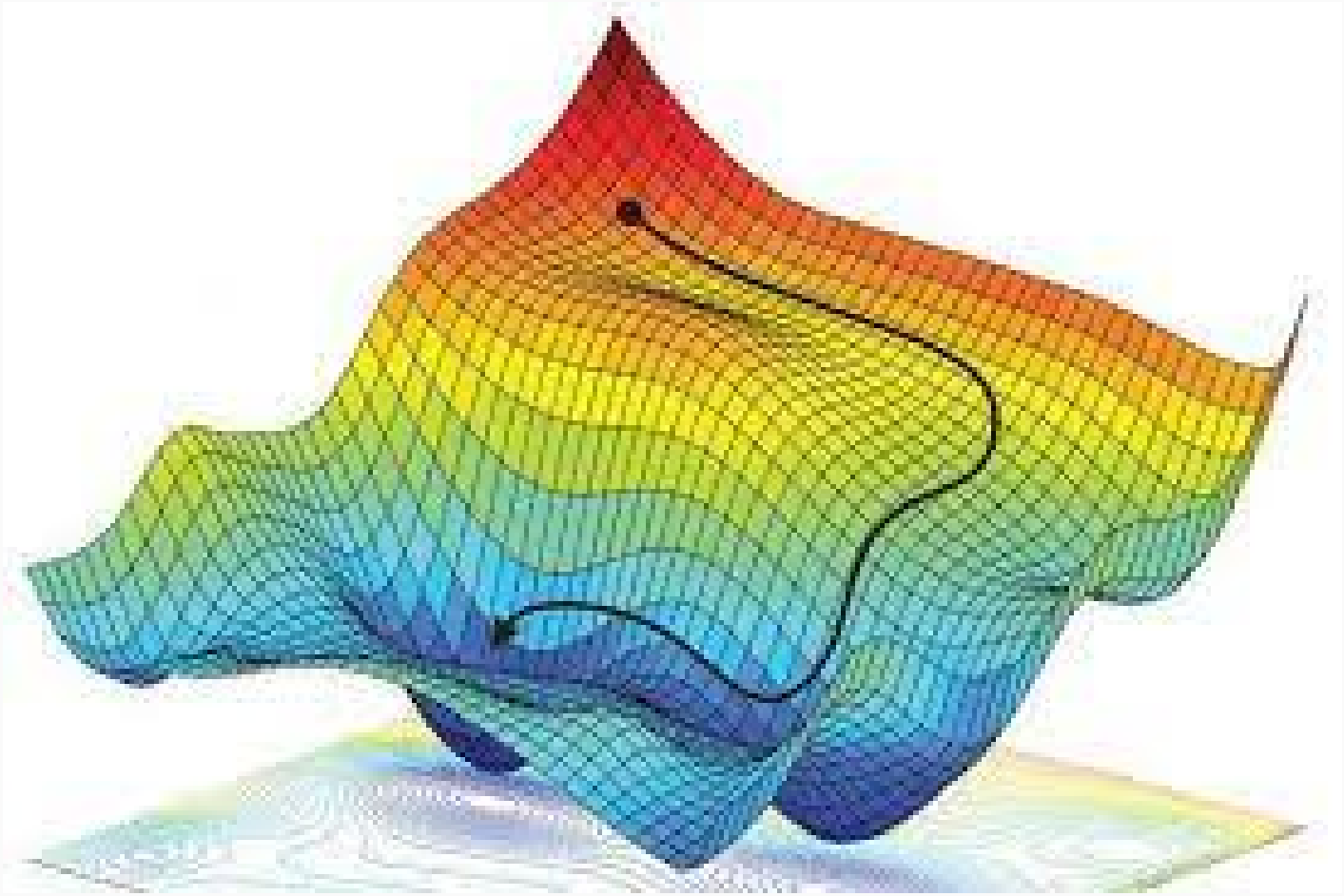

Figure 2: GD-3d

Properties of Gradient Descend Algorithm

- The Gradient Descend algorithm allows us to find a local minimal of $L_n(\beta)$. It does not guarantee global optimality.
- For linear regression, the Gradient Descend algorithm always finds the global minimum. Why? Convexity.
- The computation cost of Gradient Descend is only proportional to p, i.e., np, not p^3.
- But this could be still large when n is large.

Fast Gradient Descend Algorithms

- To compute the Gradient Descend (GD) faster, there are alternations.
- The Stochastic Gradient Descend (SGD), refers to a special kind of algorithm that updates β using only part of the data each time, not the full sample - leading to faster convergence.
- The Gradient Boosting (GB) Algorithm, refers to a special kind of algorithm that updates β using the full sample, but only through one dimension among all the p dimensions of β.

The Stochastic Gradient Descend Algorithm

- Let us call D_j as a Batch of data, $j = 1, 2, ..., K$. The union $\cup_{j=1}^{K} D_j$ is the full sample. So here we divide the full sample as subsets of sample. These sub-samples usually have the same size.
- Let B be the size of D_j. B is called batch size.
- The updating rule is changed to

$$\beta_{j+1} = \beta_j - \frac{\alpha_j}{B} \sum_{i \in D_j} L(Y_i, f(X_i, \beta_j)).$$

The SGD

- $\frac{1}{B}\sum_{i\in D_j} L(Y_i, f(X_i, \beta_j))$ is the subsample gradient, and since the sample is randomly created, its expectation should be the same as the gradient of the full sample, e.g., $\nabla L_n(\beta)$.
- SGD can be viewed as a stochastic version of GD - more noisy, but faster bias shrinkage (faster learning).
- Batch size B in practice can be chosen as a relatively small size compared to n, e.g., $10, 100, 1000$, depending on how large n is. In the extreme case, B can be set as 1.
- When you loop over all the $D_j, j = 1, 2, ..., K$, you can restart from D_1, or simply define $D_{K+j} = D_j$ to continue. We call one loop over $D_j, j = 1, 2, ..., K$ as an epoch of training. Ideally you may do as many epochs as possible. In practice we may stop at a finite number of epochs, e.g., 50.
- SGD also works better in the streaming data environment, i.e., data point comes in one by one.

Special Case: GD and SGD in Linear Regression

- In linear regression, the Loss function is

$$L_n(\beta) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - X_i^\intercal \beta)^2$$

- GD can be written as:

$$\beta_{t+1} = \beta_t - \alpha_t \frac{1}{n}\sum_{i=1}^{n} -X_i(Y_i - X_i^\intercal \beta_t).$$

- SGD can be written as:

$$\beta_{t+1} = \beta_t - \alpha_t \frac{1}{B}\sum_{i \in D_t} -X_i(Y_i - X_i^\intercal \beta_t).$$

Comparison

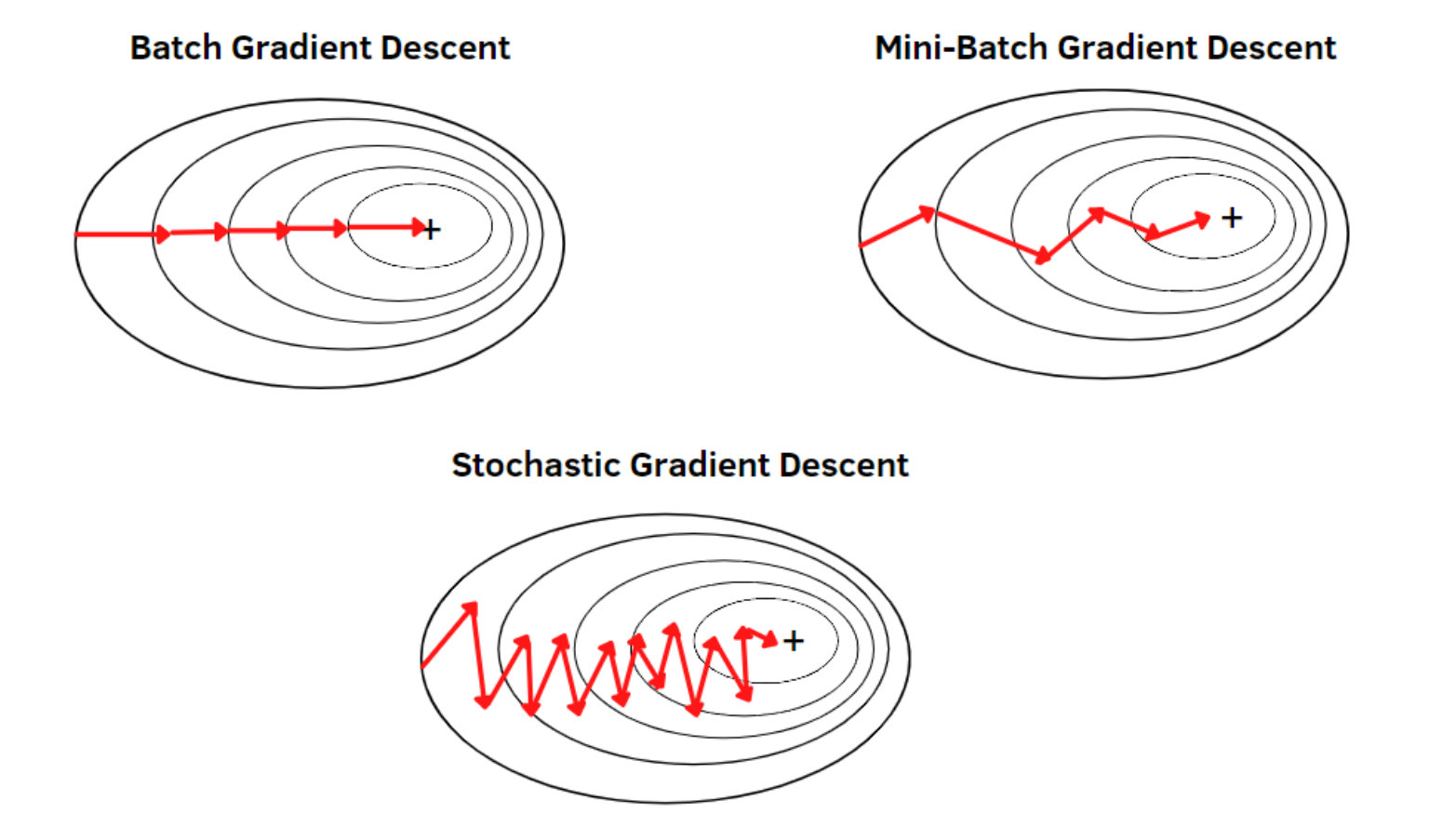

Figure 3: Comparison

Historical Remark

- AdaBoost for classification by Freund and Shapire Ensemble method, combining of many "weak" classifiers to produce a powerful "committee" (majority voting). "Best off-the-shelf classifier in the world" (Breiman, 1998) Resistance to overfitting.
- AdaBoost algorithm as gradient descent algorithm in function space (Breiman, 1998, AoS)
- Embedding of boosting algorithms into the framework of statistical estimation and additive basis expansion (Friedman, 2001, AoS and Friedman et al. 2000, AoS).
- Representation of boosting as "stagewise, additive modeling".

Boosting Algorithms in Linear Regression

- Boosting is a type of algorithm that select one out of many (p) directions for updating.
- In linear regression, the gradient on each $X_j, j = 1, 2, ..., p$ is proportional to the correlation between the two vectors X_j and Y.
- Use $< X_j, Y >$ to denote for the inner product of the two vectors, and $< X_j, Y >_n$ to denote $\frac{1}{n} < X_j, Y >$ as the sample average of $X_{ij} Y_i$.
- Normalization/standardization: Assume that all vectors X_j are demeaned to 0 and standardized with standard error being 1.

L-2 Boosting Algorithm

1. Start/initialization: $\beta^0 = 0$ (p-dimensional vector), set maximum number of iterations m_{stop} and set iteration index m to 0.
2. At the $(m+1)^{th}$ step, calculate the residuals $U_i^m = y_i - x_i^\intercal \beta^m$.
3. For each predictor variable $j = 1, \ldots, p$, calculate:

$$\gamma_j^m := \frac{\sum_{i=1}^n U_i^m x_{i,j}}{\sum_{i=1}^n x_{i,j}^2}.$$

   Select the variable j^m that is most correlated with the residuals, i.e.,

$$j^m := \text{argmax}_{1 \le j \le p} |\gamma_j^m|.$$

4. Update the estimator: $\beta^{m+1} := \beta^m + \gamma^m e_{j^m}$, where e_{j^m} is the j^mth index vector and $\gamma^m = \gamma_{j^m}^m$.
5. Increase m by one. If $m < m_{stop}$, continue with (2); otherwise stop.

What L-2 Boosting Does?

- Select one "best" predictor at a time.
- Update the residuals based on this predictor.
- Proceed to the next round, using the updated residual.
- Build models from a bottom up approach. Model Selection like AIC/BIC: build models from a top down approach.

Related Literatures

- Bühlmann and Yu (2003): Low-dimensional setting; exponential bias-variance trade-off for boosting.
- Zhang and Yu (2005): Results for convergence, consistency and statistical rates of convergence of boosting with early stopping in a low-dimensional setting.
- Bühlmann (2006): Consistency of L-2 Boosting in a high-dimensional setting.
- Luo and Spindler (2022): established convergence rate and inference for high dimensional L-2 Boosting.

Key Heuristics

- Key Quantities:
  1. $U^m := Y - X\beta^m$, where m is the number of iteration in the algorithm, β^m is the parameter β at m^{th} iteration.
  2. $\alpha^m = \beta - \beta^m$ represents the bias of the model.
  3. $V^m = X\alpha^m$ is the vector of prediction error.
- The deterministic part requires m to be large to reduce bias ($\|V^m\|$ becomes small).
- To avoid over fitting requires m to be small. Trade-off to obtain optimal convergence rates and optimal stopping time.
- Therefore, we need a Stopping Criteria that stops the algorithm in the middle. This is similar to penalization (AIC, BIC, LASSO, etc.) that puts restrictions on the size of models. (Why?)

Post-Selection Approaches

- Post LASSO: let $\hat{T}$ be the subset of $\{1, 2, ..., p\}$ such that $\hat{T} = \{j | \beta_{LASSO,j} \neq 0\}$.
  Then, we define post-LASSO as

$$\beta_{Post-LASSO} = argmin_{\beta, \beta_{\hat{T}^c}=0} \sum_{i=1}^{n} (Y_i - X_i\beta)^2.$$

- Similarly, for Boosting, we can define $\hat{T}_B$ as the variables selected by Boosting before being stopped.
- The post-Boosting estimator can be defined as:

$$\beta_{Post-L2Boosting} = argmin_{\beta, \beta_{\hat{T}_B^c}=0} \sum_{i=1}^{n} (Y_i - X_i\beta)^2.$$

A similar algorithm: the orthogonal Boosting

1. Start/initialization: $\beta_o^0 = 0$ (p-dimensional vector), set maximum number of iterations m_{stop} and set iteration index m to 0.
2. At the $(m+1)^{th}$ step, calculate the residuals $U_i^m = y_i - x_i^\intercal \beta_o^m$, for $i = 1, 2, \ldots, n$.
3. For each predictor variable $j = 1, \ldots, p$, calculate:

$$\gamma_j^m := \frac{\sum_{i=1}^n U_i^m x_{i,j}}{\sum_{i=1}^n x_{i,j}^2}.$$

   Select the variable j^m that is most correlated with the residuals, i.e.,

$$j^m := \text{argmax}_{1 \le j \le p} |\gamma_j^m|.$$

4. Update the estimator: $\beta_o^{m+1} := (X_{T_o^{m+1}}{}^\intercal X_{T_o^{m+1}})^{-1} X_{T_o^{m+1}}{}^\intercal Y$, with $T_o^{m+1} := \cup_{t=0}^m \{j^t\}$.
5. Increase m by one. If $m < m_{stop}$, continue with (2); otherwise stop.

The Stopping Criteria

- Simple idea: should not stop too early nor too late.
- An infeasible criteria: stop at Ks, where s is the size of the "true" model, and K is a constant. Another one: the oracle criteria: stop at where out-of-sample MSE is the minimum.
- Of course, you can always use CV.

Stopping Criteria in L2-Boosting

- Peter Bulmann's BIC approach: At each m, evaluate $BIC_m = \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i\beta^m)^2 + 2m\log n$. Select m that minimizes BIC_m. Guarantee convergence but may lead to slow convergence rate.
- Luo and Spindler's (2022 JOE) data driven approach: Compare the size between residuals U^{m+1} and U^m. Stop when:

$$\frac{\|U^{m+1}\|^2}{\|U^m\|^2} > 1 - 4.4\frac{\log 2p}{n}.$$

Simulation Experiments

- Suppose the model follows a linear functional form with large p:

$$y_i = x_i\beta + \epsilon_i,$$

  with $\beta = (\beta_1, \beta_2, ..., \beta_p)$.
- Two sets of settings are considered:
  1. Sparse: $\beta = (1, 1, ..., 1, 0, 0, ..., 0)$. A few components are 1, others are all 0.
  2. Approximate Sparse: β follows a decay pattern, e.g., $\beta = (1, 1/2, 1/3, ..., 1/p)$.

Simulation Settings

For the design matrix X, we consider the following two settings.

- "orthogonal" setting: entries of X are drawn as i.i.d. draws from a standard normal distribution.
- "correlated" setting: X_i (rows of X) are distributed according to a multivariate normal distribution where the correlations are given by a Toeplitz matrix with factor 0.5 and alternating sign.
- $n = 100, 200, 400, 800$, $p = 100, 200$. s : number of non-zero components in the sparse case, $s = 10$ in the simulation.
- Out-of-Sample size: $n_1 = 50$. The experiment is repeated for 500 times.

I.i.d. Sparse Case

n	p	s	PGA-oracle	PGA-Ks	PGA-our	post-PGA-oracle	post-PGA-Ks	post-PGA-our
100	100	10	0.475	0.713	0.624	0.121	0.566	0.335
100	200	10	0.581	0.953	0.804	0.127	0.733	0.443
200	200	10	0.184	0.367	0.295	0.054	0.313	0.195
400	200	10	0.084	0.173	0.144	0.026	0.145	0.103
800	200	10	0.036	0.079	0.065	0.013	0.067	0.048

n	p	s	OGA-oracle	OGA-Ks	OGA-our
100	100	10	0.118	0.802	0.485
100	200	10	0.124	1.002	0.715
200	200	10	0.054	0.424	0.260
400	200	10	0.026	0.197	0.128
800	200	10	0.013	0.090	0.057

n	p	s	LASSO	post-LASSO	Lasso-CV	post-Lasso-CV
100	100	10	0.897	0.606	0.540	0.877
100	200	10	1.209	1.335	0.758	1.135
200	200	10	0.348	0.416	0.276	0.510
400	200	10	0.141	0.188	0.119	0.270
800	200	10	0.067	0.088	0.056	0.135

Figure 4: MSE of different methods in i.i.d. Sparse case

Correlated Sparse Case

n	p	s	PGA-oracle	PGA-Ks	PGA-our	post-PGA-oracle	post-PGA-Ks	post-PGA-our
100	100	10	1.382	1.651	2.134	0.447	0.846	1.704
100	200	10	3.112	3.034	3.121	2.198	2.498	2.953
200	200	10	0.631	0.735	0.811	0.060	0.174	0.167
400	200	10	0.199	0.259	0.247	0.025	0.038	0.087
800	200	10	0.079	0.207	0.102	0.014	0.014	0.042

n	p	s	OGA-oracle	OGA-Ks	OGA-our
100	100	10	0.279	0.743	1.0
100	200	10	3.026	2.580	2.679
200	200	10	0.057	0.390	0.228
400	200	10	0.025	0.189	0.115
800	200	10	0.014	0.086	0.052

n	p	s	LASSO	post-LASSO	Lasso-CV	post-Lasso-CV
100	100	10	2.896	1.271	0.830	1.201
100	200	10	3.115	2.260	1.874	2.422
200	200	10	1.529	0.392	0.466	0.758
400	200	10	0.415	0.170	0.192	0.356
800	200	10	0.176	0.079	0.088	0.171

Figure 5: MSE of different methods in correlated Sparse case

I.i.d. Poly-Decay Case

n	p	s	PGA-oracle	PGA-Ks	PGA-our	post-PGA-oracle	post-PGA-Ks	post-PGA-our
100	100	10	0.398	0.772	0.575	0.392	0.959	0.616
100	200	10	0.457	1.021	0.673	0.445	1.319	0.728
200	200	10	0.274	0.513	0.364	0.269	0.602	0.382
400	200	10	0.199	0.276	0.240	0.194	0.294	0.244
800	200	10	0.125	0.149	0.147	0.125	0.152	0.149

n	p	s	OGA-oracle	OGA-Ks	OGA-our
100	100	10	0.413	1.148	0.719
100	200	10	0.440	1.574	0.862
200	200	10	0.269	0.655	0.435
400	200	10	0.191	0.308	0.257
800	200	10	0.126	0.155	0.152

n	p	s	LASSO	post-LASSO	Lasso-CV	post-Lasso-CV
100	100	10	0.436	0.653	0.428	0.709
100	200	10	0.504	1.170	0.551	0.877
200	200	10	0.325	0.521	0.311	0.518
400	200	10	0.212	0.294	0.190	0.343
800	200	10	0.145	0.166	0.118	0.213

Figure 6: MSE of different methods in iid non-Sparse case

Correlated Poly-Decay Case

n	p	s	PGA-oracle	PGA-Ks	PGA-our	post-PGA-oracle	post-PGA-Ks	post-PGA-our
100	100	10	0.231	0.648	0.419	0.217	0.864	0.434
100	200	10	0.261	0.896	0.506	0.255	1.207	0.534
200	200	10	0.195	0.482	0.289	0.169	0.566	0.271
400	200	10	0.146	0.258	0.204	0.119	0.270	0.180
800	200	10	0.105	0.139	0.126	0.083	0.132	0.110

n	p	s	OGA-oracle	OGA-Ks	OGA-our
100	100	10	0.215	1.052	0.524
100	200	10	0.255	1.538	0.657
200	200	10	0.172	0.619	0.308
400	200	10	0.114	0.294	0.185
800	200	10	0.077	0.142	0.109

n	p	s	LASSO	post-LASSO	Lasso-CV	post-Lasso-CV
100	100	10	0.316	0.569	0.339	0.585
100	200	10	0.354	0.877	0.412	0.658
200	200	10	0.263	0.426	0.269	0.430
400	200	10	0.184	0.245	0.170	0.304
800	200	10	0.134	0.141	0.107	0.191

Figure 7: MSE of different methods in correlated non-Sparse case

Out-of-Sample MSE

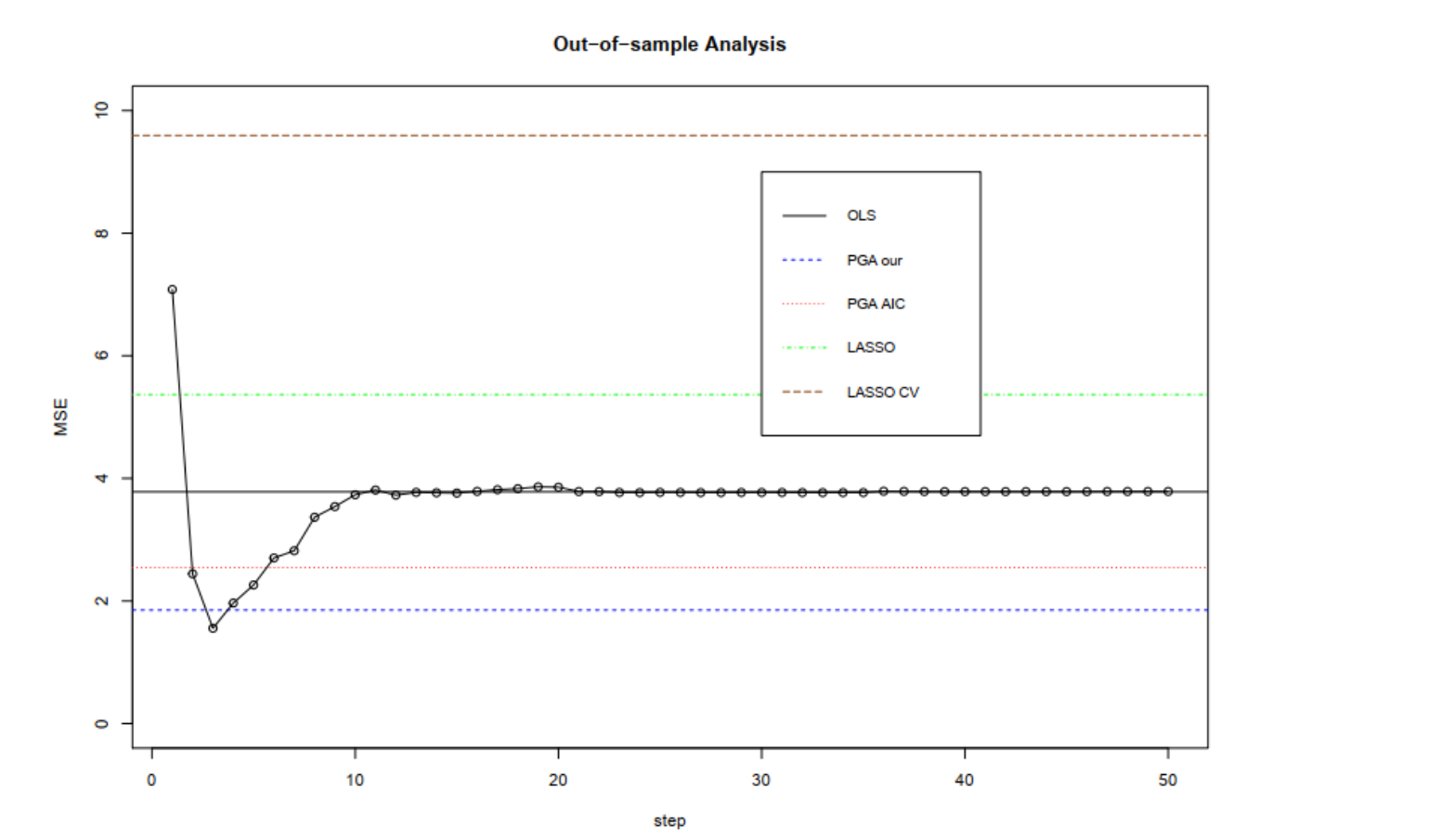

Figure 8: Out-of-Sample MSE

Computation Time Comparison

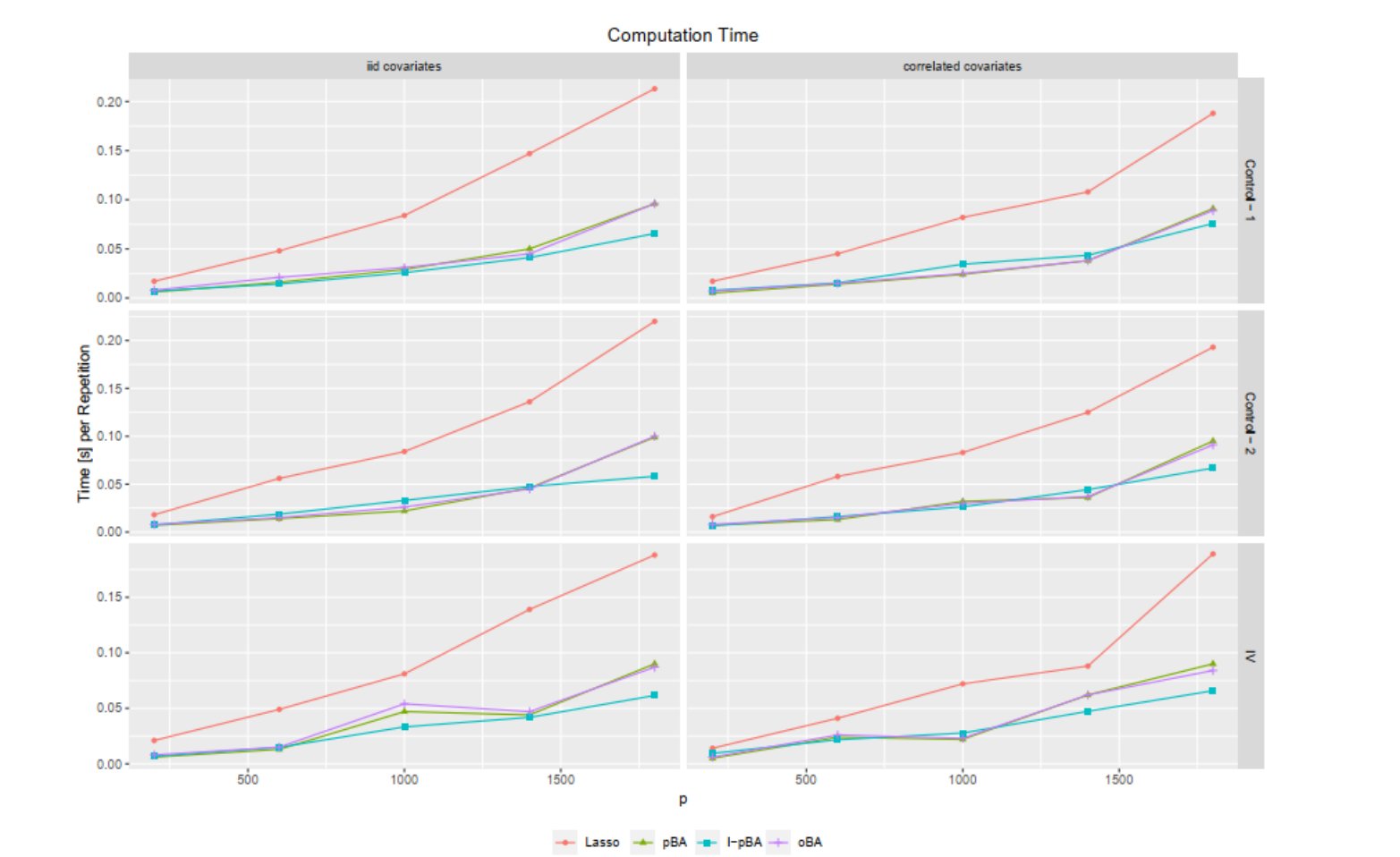

Figure 9: Computational Time of Different Algos

Application: Empirical Analysis on Bank Mergers

- We use a data set from Levine, Lin and Wang (2020), who study how the pre-merger overlap of bank branch networks in the US can affect the postmerger stock returns of acquirer banks over the period 1984 to 1995. The target variable of interest is the cumulative abnormal returns (CAR) of the acquirer bank within a window of five days around the announcement of the acquisition, denoted CAR(-2,2)
- We are interested in estimating the causal effect of the overlap of bank branch networks on the outcome of the acquisition, i.e., CAR(-2,2). This overlap is measured by three variables, namely the Overlap, Correlation, and Cosine Distance.

Data And Variables

- The Overlap variable is measured as the quotient of the number of states in which both banks are present and the total number of states in which at least one bank is present.
- Correlation and Cosine Distance are derived from the vector for each bank, which denotes if they are present in each state.

Key problems in financial economics

- The endogeneity problem arises because measures of the overlap of bank branch networks and bank merger decisions and outcomes may be affected by unobservable hidden factors.
- In many states in the US, banks were not allowed to set up branches in other states. In the 1980s and 1990s, states gradually relaxed this restriction, which was largely exogenous to the individual banks (Jayaratne and Strahan (1996)).
- It is common to utilize neighborhood structure of nodes in a network to serve as instruments to deal with endogeneity in the literature, e.g., Chandrasekhar (2016). We define $d(S_1, S_2)$ as the minimum number of edges that links S_1, S_2.
- Note that the state network has a spatial structure. Consider the case that a bank (the acquirer) in S_1 acquires a bank in S_2. Consider measuring "overlap" in the state network for two states S_1; S_0 relative to the target bank's state, S_2.
- We construct the "connection measure" between two banks S_1, S_2 by considering their geographical distance weighted overlaps of branches.

Result Table

TABLE 2. Second stage estimates using iterated post- and orthogonal L_2-Boosting.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
	Dependent Variable: Acquirer CAR (-2, +2)											
	Original IV			iterated post-L_2-Boosting			orthogonal L_2-Boosting			post-Lasso		
Overlap	17.18***			8.827**			8.753**			3.575		
	(5.27)			(3.53)			(3.52)			(4.39)		
Correlation Coefficient		12.93***			11.50***			12.34***			5.102	
		(4.86)			(4.44)			(4.68)			(4.73)	
Cosine Distance			-17.66***			-13.64***			-8.386			-5.364
			(5.63)			(5.28)			(5.12)			(4.84)
Firm-year Controls	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Year Fixed Effects	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Acquirer-State × Target-State Fixed Effects	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Observations	442	442	442	442	442	442	442	442	442	442	442	442
R-squared	0.088	0.298	0.144	0.404	0.338	0.288	0.406	0.315	0.410	0.466	0.446	0.446
Algorithm Execution Time (ms)				3.313	2.381	2.426	3.658	3.866	4.227	19.928	5.260	5.298

TABLE 3. Selected instrumental variables that are used in Table 2.

Column	Instrumental Variables
(1)	Original SNN IV for Overlap
(2)	Original SNN IV for Correlation Coefficient
(3)	Original SNN IV for Cosine Distance
(4)	$FNN1_{DC}, FNN1_{IA}, FNN1_{VA}, FNN2_{MD}, FNN2_{NJ}, FNN2_{KS}$
(5)	$FNN1_{AR}, FNN1_{VA}, FNN2_{MD}, FNN3_{KS}$
(6)	$FNN1_{AR}, FNN2_{TX}, FNN3_{KS}, FNN3_{NY}, FNN3_{RI}$
(7)	$FNN1_{AR}, FNN1_{DC}, FNN1_{IA}, FNN1_{VA}, FNN2_{MD}, FNN2_{NJ}$
(8)	$FNN1_{AR}, FNN1_{IA}, FNN1_{VA}, FNN2_{MD}, FNN3_{KS}$
(9)	$FNN1_{AR}, FNN1_{IA}, FNN2_{OK}, FNN2_{TX}, FNN3_{KS}, FNN3_{RI}$
(10)	$FNN2_{MI}, FNN2_{NY}, FNN3_{AK}, FNN3_{LA}$
(11)	$FNN2_{MI}, FNN2_{NY}, FNN3_{LA}$
(12)	$FNN2_{MI}, FNN2_{NY}, FNN2_{PA}, FNN3_{LA}$