Collapsing-Fast-Large-Almost-Matching-Exactly: A Matching Method for Causal Inference

Awa Dieng[†], Yameng Liu[†], Sudeepa Roy[†], Cynthia Rudin[†‡¶], Alexander Volfovsky[¶*]

[†]Department of Computer Science
[‡]Department of Electrical and Computer Engineering
[¶]Department of Statistical Science
Duke University
{adieng, ymliu, sudeepa, cynthia}@cs.duke.edu, alexander.volfovsky@duke.edu

April 17, 2021

Abstract

We aim to create the highest possible quality of treatment-control matches for categorical data in the potential outcomes framework. Matching methods are heavily used in the social sciences due to their interpretability, but most matching methods in the past do not pass basic sanity checks in that they fail when irrelevant variables are introduced. Also, past methods tend to be either computationally slow or produce poor matches. The method proposed in this work aims to match units on a weighted Hamming distance, taking into account the relative importance of the covariates; the algorithm aims to match units on as many relevant variables as possible. To do this, the algorithm creates a hierarchy of covariate combinations on which to match (similar to downward closure), in the process solving an optimization problem for each unit in order to construct the optimal matches. The algorithm uses a single dynamic program to solve all of optimization problems simultaneously. Notable advantages of our method over existing matching procedures are its high-quality matches, versatility in handling different data distributions that may have irrelevant variables, and ability to handle missing data by matching on as many available covariates as possible.

Keywords: Causal inference, matching

1 Introduction

In observational causal inference where the scientist does not control the randomization of individuals into treatment, an ideal approach matches each treatment unit to a control unit with identical covariates. However, in high dimensions, few such "identical twins" exist, since it becomes unlikely that any two units have identical covariates in high dimensions. In that case, how might we construct a match assignment that would lead to accurate estimates of conditional average treatment effects (CATEs)?

*This work was supported in part by NIH award 1R01EB025021-01, NSF awards IIS-1552538 and IIS-1703431, a DARPA award under the L2M program, and a Duke University Energy Initiative Energy Research Seed Fund (ERSF). All authors contributed equally to this work.

For categorical variables, we might choose a Hamming distance to measure similarity between covariates. Then, the goal is to find control units that are similar to the treatment units on as many covariates as possible. This leads to two challenges, the first being computational (how does one compute optimal matches on Hamming distance?), and the second being that not all covariates are equally relevant, which can impact the Hamming distance.

This second issue, that not all covariates are equally important, has serious implications for CATE estimation quality. Matching methods generally suffer when many irrelevant covariates (covariates that are not related to either treatment or outcome) are introduced. In particular, the irrelevant variables would dominate the Hamming distance calculation, so that the treatment units would mainly be matched to the control units on the irrelevant variables. This means that matching methods do not always pass an important sanity check: irrelevant variables should be irrelevant. As our experiments show, the participation of irrelevant variables can overwhelm many state of the art matching methods, including Propensity Score Matching [12], Nearest Neighbor matching [14], as well as methods not explicitly designed for matching such as [22] Causal Forest.

We propose a method in this work that aims to solve both of the problems listed above. The algorithm uses a hold-out training set to determine which variables are the most important to match on. It then determines matches by optimizing a weighted Hamming distance, where the weights on each variable are determined from the analysis of the hold-out set. Finding a matched group for a given unit then becomes a constrained discrete optimization problem, and we solve all of these optimization problems efficiently with a single dynamic program. This algorithm has the same basic monotonicity property (downwards closure) and structure as that of the apriori algorithm [1] used in data mining for finding frequent itemsets. However, frequency of itemsets is irrelevant here, instead the goal is to find a largest (weighted) number of covariates that both a treatment and control unit have in common. The algorithm fully optimizes the weighted Hamming distance of each treatment unit to the nearest control unit (and vice versa). It is efficient, owing to the use of database programming and bit-vector computations, and does not require an integer programming solver.

The method is named Collapsing Fast Large Almost Matching Exactly (Collapsing FLAME), and extends the FLAME algorithm of [13]. Basic FLAME does not optimize Hamming distance, it uses computationally efficient backwards elimination. Collapsing FLAME instead considers all reasonable subsets of covariates by eliminating variables, adding them back in, and then eliminating other variables (collapsing and expanding the set of variables). Basic FLAME does not use the downward closure property to choose covariates; its backwards elimination method is extremely fast, but it cannot produce the high-quality matches that Collapsing FLAME produces. Collapsing FLAME's matched groups tend to match on more covariates than the basic FLAME algorithm; the distances between matched units

in collapsing FLAME are smaller within each matched group, thus its matches are distinctly higher quality. This has implications for missing data, where Collapsing FLAME can find matched groups that FLAME cannot.

2 Related Work

Randomized experiments are a gold standard for causal inference [18]; however, much of the available data for causal inference is observational, where the experimenter does not control the randomization mechanism. Observational datasets can be much larger and easier to obtain than randomized trial data, and matching techniques provide ways to approximate the covariate balance that would have come from a randomized trial.

Exact matching is not possible in high dimensions, as "identical twins" in treatment and control samples are not likely to exist. Early on, this led to techniques that reduce dimension using propensity score matching [15], [14], [16], [4], which extend to penalized regression approaches [20], [10], [3], [5]. Propensity score matching methods project the entire dataset to one dimension and thus cannot be used for estimating CATE (conditional average treatment effect), since the matched groups often do not agree on important covariates. In "optimal matching," [11], an optimization problem is formed to choose matches according to a pre-defined distance measure, with balance constraints, though as discussed above, this distance measure can be dominated by irrelevant covariates, often leading to poor matched groups and biased estimates of treatment effect. Coarsened exact matching [8, 9] has the same problem.

A recent alternative framework is that of *almost-exact matching*, where each matched group contains units that are close on covariates that are important for predicting outcomes.
For example, Coarsened Exact Matching [8, 9] is almost-exact, if one were to use an oracle (should one ever become available) that bins covariates according to importance for estimating causal effects. The original FLAME algorithm [13] is an almost-exact matching method that adapts the distance metric to the data using machine learning. It starts by matching "identical twins," and proceeds by eliminating less important covariates one by one, attempting to match individuals on the largest set of covariates available. FLAME has a balance constraint to ensure that it does not remove too many treatment or control units as it eliminates covariates. It uses a hold-out training set to determine importance of covariates. The combination of balance and importance of covariates is called "match quality" (MQ). FLAME can handle datasets too large to fit in memory, and scales well with the number of covariates, but removing covariates in exactly one order (rather than all possible orders as in the present work) means that many high-quality matches will be missed.

3 Almost-Exact Matching Framework

Consider a dataframe $D = [X, Y, T]$ where $X \in \mathbb{R}^{n \times p}$, $Y \in \mathbb{R}^n$, $T \in \{0,1\}^n$ respectively denote the covariates for all units, the outcome vector and the treatment indicator (1 for treated, 0 for control). Unit i's jth covariate X is denoted $x_{ij} \in \{0,1\}$. Notation $\mathbf{x}_i \in \mathbb{R}^p$ indicates covariates for the ith unit, and $T(i) \in \{0,1\}$ is an indicator for whether or not unit i is treated.

Throughout we make SUTVA and ignorability assumptions [17]. The goal is to match treatment and control units on as many relevant covariates as possible. Relevance of covariate j is denoted by $w_j \geq 0$ and it is determined using a hold-out training set. Either w_j's can be fixed beforehand or adjusted dynamically using a for-loop inside the algorithm as FLAME does.

Assuming that we have a fixed weight w_j for each covariate j, we would like to find a match for each treatment unit t that *matches at least one control unit on as many relevant covariates as possible*. Thus we consider the following matching problem:

Almost-Exact Matching with Replacement (AEMR) *For each treatment unit t,*

$$\mathbf{v}^{t*} \in \text{argmax}_{\mathbf{v} \in \{0,1\}^p} \mathbf{v}^T \mathbf{w} \textit{ such that } \quad \exists\, \ell \;\; \textit{with } T(\ell) = 0 \textit{ and } \mathbf{x}_\ell \circ \mathbf{v} = \mathbf{x}_t \circ \mathbf{v},$$

where $\circ$ denotes Hadamard product. The solution to the AEMR problem is an indicator of the optimal set of covariates for treatment unit t's matched group. The constraint says that the optimal matched group contains at least one control unit.

The solution of the AEMR problem can be the same for multiple treatment units, in which case they would form a single matched group. Formally, we define the matched group for treatment unit t.
The **main matched group** corresponding to treatment unit t contains all units ℓ such that

$$\mathbf{x}_t \circ \mathbf{v}^{t*} = \mathbf{x}_\ell \circ \mathbf{v}^{t*}.$$

That is, the main matched group contains all treatment and control units that have identical covariate values to t on the covariates for which $\mathbf{v}^{t*}$ is 1. The formulation of the AEMR and main matched group formulation is symmetric for control units.

To compute the solution to the AEMR problem, as well as all main matched groups, we will leverage efficient database queries for exact matching, in the `GroupedExactMatch` subroutine in Algorithm 1. Group-by queries can be performed efficiently on data that are too large to fit in memory. An efficient bit-vector implementation is used instead when data are able to fit in memory.

`GroupedExactMatch` takes a given subset of covariates and finds all subsets of treatment and control units that have identical values of those covariates. The `GroupedExactMatch` subroutine is called repeatedly in our `Collapsing-FLAME` algorithm (described in the next section) for different subsets of covariates. Details on group-by and prune are available in the appendix.

Algorithm 1: Procedure `GroupedExactMatch`

Input : Unmatched Data $D^{um} = (X, Y, T)$, subset of indexes of covariates $J^s \subseteq \{1, ..., p\}$.

Output : Newly matched units D^m using covariates indexed by J^s where groups have at least one treatment and one control unit, and the matched groups for D^m.

M_{raw}=`group-by` (D^{um}, J^s) (`form groups by exact matching on` J^s)

M=`prune(`M_{raw}`) (remove groups without at least one treatment and control unit)`

D^m=`Get subset of` D^{um} `where the covariates match with` M `(recover newly matched units)`

return $\{D^m, M\}$. *(*`matched units and matched groups`*)*

There are straightforward (but inefficient) approaches to solving the AEMR problem for all units.

AEMR Solution 1 (quadratic in n, linear in p): Brute force pairwise comparison of treatment points to control points. For all treatment units t, we (i) iterate over all control units c, (ii) find the vector $\mathbf{v}_{tc} \in \{0, 1\}^p$ with value 1 if there is a match on the values of the corresponding covariates, and 0 otherwise, (iii) find the control unit(s) with the highest value of $\mathbf{v}_{tc}^T \mathbf{w}$, and (iv) return them as the main matched group for the treatment unit t (and compute the auxiliary group). Whenever a previously matched unit α is matched to a previously unmatched unit η, record the η's main matched group as an auxiliary group for the previously matched unit α. When all units are 'done' (all units are either matched already or cannot be matched) then stop, and compute the CATE for each treatment and control unit using its main matched group. If a unit belongs to auxiliary matched groups then its outcome is used for computing both its own CATE (in its own main matched group) and the CATEs of units for whom it is in an auxiliary group (e.g., α will be used to compute η's estimated CATE). This algorithm is polynomial in both n and p, however, the quadratic time complexity in n also makes this approach impractical for large datasets (for instance, when we have more than a million units with half being treatment units).

AEMR Solution 2 (order $n \log n$, exponential in p): Brute force iteration over all 2^p subsets of the p covariates. This approach solves the AEMR problem simultaneously for all treatment and control units for a fixed weight vector $\mathbf{w}$. First, (i) enumerate every $\mathbf{v} \in \{0, 1\}^p$ (which serves as an indicator for a subset of covariates), (ii) order the $\mathbf{v}$'s according to $\mathbf{v}^T \mathbf{w}$, (iii) call `GroupedExactMatch` for every $\mathbf{v}$ in the predetermined order, (iv) the first time each unit is matched during a `GroupedExactMatch`

procedure, mark that unit with a 'done' flag, and record its corresponding main matched group and, to facilitate matching with replacement, (v) whenever a previously matched unit is matched to a previously unmatched unit, record this main matched group as an auxiliary group. When all units are 'done' (all units are either matched already or cannot be matched) then stop, and compute the CATE for each treatment and control unit using its main matched group. Each unit's outcome will be used to estimate CATEs for every auxiliary group that it is a member of, as before. Although this approach exploits the efficient 'group by' function (e.g., provided in database (SQL) queries), which can be implemented in $O(n \log n)$ time by sorting the units, iterating over all possible vectors $\mathbf{v} \in \{0, 1\}^p$ makes this approach unsuitable for practical purposes (exponential in p).

If n is in the millions, the first solution, or any simple variation of it, is practically infeasible. However, even if p is reasonably large (hundreds, thousands), a monotonicity property (downward closure) will allow us to prune the search space so that the second solution can be modified to be completely practical, as we will demonstrate in `Collapsing-FLAME` which does not enumerate all $\mathbf{v}$'s, it computes them dynamically.

Proposition 3.1. (Monotonicity of $\mathbf{v}^*$ in AEMR solutions) *Fix treatment unit t. Consider feasible* $\mathbf{v}$*, meaning* $\exists\ \ell$ *with* $T(\ell) = 0$ *and* $\mathbf{x}_\ell \circ \mathbf{v} = \mathbf{x}_t \circ \mathbf{v}$*. Then,*

- *Any feasible* $\mathbf{v}'$ *such that* $\mathbf{v}' < \mathbf{v}$ *elementwise will have* $\mathbf{v}'^T\mathbf{w} \leq \mathbf{v}^T\mathbf{w}$.
- *Consequently, consider feasible vectors* $\mathbf{v}$ *and* $\mathbf{v}'$*. Define* $\tilde{\mathbf{v}}$ *as the elementwise* $\min(\mathbf{v}, \mathbf{v}')$*. Then* $\tilde{\mathbf{v}}^T\mathbf{w} < \mathbf{v}^T\mathbf{w}$*, and* $\tilde{\mathbf{v}}^T\mathbf{w} < \mathbf{v}'^T\mathbf{w}$.

These follow from the fact that the elements of $\mathbf{v}$ are binary and the elements of $\mathbf{w}$ are non-negative. The first property means that if we have found a feasible $\mathbf{v}$, we need not consider any $\mathbf{v}'$ with fewer 1's as a possible solution of the AEMR for unit t. Thus, `Collapsing-FLAME` starts from $\mathbf{v}$ being all 1's (consider all covariates). It systematically drops one element of $\mathbf{v}$ to zero at a time, then two, then three, ordered according to values of $\mathbf{v}^T\mathbf{w}$. The second property implies that we must evaluate both $\mathbf{v}$ and $\mathbf{v}'$ as possible AEMR solutions before evaluating $\tilde{\mathbf{v}}$. Conversely, a new subset of variables defined by $\tilde{\mathbf{v}}$ cannot be considered unless all of its supersets have been considered.

We now have ingredients for `Collapsing-FLAME`: the dynamic programming mechanism over subsets of variables and fast implementations of `GroupedExactMatch` using database queries and bit vectors.

4 Algorithm: Collapsing FLAME

AEMR has two computations, the weights and the actual matching mechanism. The weights correspond to the importance of the covariates and are computed by running regression on a hold-out training set

prior to running the algorithm (or computed adaptively, the computation of the weights is the same as in the FLAME algorithm [13] and is discussed in the appendix). Our contribution lies in solving the AEMR and constructing the matches for all treatment and control points.

We call a *covariate-set* any set of covariates. We denote by $\mathcal{J}$ the original set of all covariates from the input dataset, where $p = |\mathcal{J}|$. When we *drop* a set of covariates s, it means we will match on $\mathcal{J} \setminus s$. For any covariate-set s, we associate an **indicator-vector** $\mathbf{v}_s \in \{0,1\}^p$ defined as follows:

$$\mathbf{v}_{s,i} = \mathbb{1}_{\{i \notin s\}} \quad \forall\, i \in \{1, .., p\} \tag{1}$$

i.e. the value is 1 if the covariate is *not in s* implying that it is being used for matching.
Algorithm 2 gives the pseudocode of the `Collapsing-FLAME` algorithm.

Algorithm 2: The `Collapsing-FLAME` algorithm

Input : Data D, precomputed weight vector w for all covariates (see appendix)

Output : $\{D^m_{(i)}, \mathcal{MG}_{(i)}\}_{i \geq 1}$ all matched units and all the matched groups from all iterations i

Notation: i= iterations, $D_{(i)}$ ($D^m_{(i)}$) = unmatched (matched) units at the end of iteration i, $\mathcal{MG}_{(i)}$ = matched groups at the end of iteration i, $\Lambda_{(i)}$ = set of active covariate-sets at the end of iteration i that are eligible be dropped to form matched groups, $\Delta_{(i)}$ = set of covariate-sets at the end of iteration i that have been processed (considered for dropping and formulation of matched groups).

Initialize: $D_{(0)} = D, D^m_{(0)} = \emptyset, \mathcal{MG}_{(0)} = \emptyset, \Lambda_{(0)} = \{\{1\}, ..., \{p\}\}, \Delta_{(0)} = \emptyset, i = 1$

while *there is at least one treatment unit to match in* $D_{(i-1)}$ **do**

- `(find the 'best' covariate-set to drop from the set of active covariate-sets)`
- Let $s^*_{(i)} = \arg\max_{s \in \Lambda_{i-1}} \mathbf{v}_s^T \mathbf{w}$ (`` `v`$_s$ denotes the indicator-vector of $s$ as in (1) ``)
- $(D^m_{(i)}, \mathcal{MG}_{(i)})$ = `GroupedExactMatch`$^*(D, D_{(i-1)}, J \setminus s^*_{(i)})$ `(find matched units and main groups)`
- $Z_{(i)}$ = `GenerateNewActiveSets`$(\Delta_{(i-1)}, s^*_{(i)})$ `(find new active covariate-sets)`
- $\Lambda_{(i)} = \Lambda_{(i-1)} \setminus \{s^*_{(i)}\}$ `(remove` $s^*_{(i)}$ `from the set of active sets)`
- $\Lambda_{(i)} = \Lambda_{(i)} \cup Z_{(i)}$ `(update the set of active sets)`
- $\Delta_{(i)} = \Delta_{(i-1)} \cup \{s^*_{(i)}\}$ `(update the set of already processed covariate-sets)`
- $D_{(i)} = D_{(i-1)} \setminus D^m_{(i-1)}$ `(remove matches)`
- $i = i + 1$

return $\{D^m_{(i)}, \mathcal{MG}_{(i)}\}_{i \geq 1}$

Instead of looping over all possible $2^{|\mathcal{J}|}$ vectors to solve the AEMR, it considers a covariate-set s for dropping only if satisfies the monotonicity property of Proposition 3.1. For example, if $\{1\}$ has been considered for dropping to form matched groups, it would not process $\{1, 2, 3\}$ next because the monotonicity property requires $\{1, 2\}$, $\{1, 3\}$ and $\{2, 3\}$ to have been considered previously for dropping.

`Collapsing-FLAME` keeps track of two sets of covariate-sets: (1) The set of **processed sets** Δ contains the covariate-sets whose main matched groups (if any exist) have already been formed. That is, Δ contains s if matches have been constructed on $\mathcal{J} \setminus s$ by calling the `GroupedExactMatch` procedure. (2) The set of **active sets** Λ contains the covariate-sets s that are eligible to be dropped according to Proposition 3.1. For any iteration i, $\Lambda_{(i)} \cap \Delta_{(i)} = \emptyset$, i.e., the sets are disjoint, where $\Lambda_{(i)}, \Delta_{(i)}$ denote the states of Λ, Δ at the end of iteration i. Due to the monotonicity property Proposition 3.1, if $s \in \Lambda_{(i)}$, then each proper subset $r \subset s$ belonged to $\Lambda_{(j)}$ in an earlier iteration $j < i$. Once an active set $s \in \Lambda_{(i-1)}$ is chosen as the optimal subset to drop $s^*_{(i)}$ in iteration i, s is excluded from $\Lambda_{(i)}$ (it is no longer active) and is included in $\Delta_{(i)}$ as a processed set. In that sense, the active sets are generated and included in $\Lambda_{(i)}$ in a hierarchical manner similar to the apriori algorithm. A set s is included in $\Lambda_{(i)}$ only if all of its proper subsets of one less size $r \subset s$, $|r| = |s| - 1$, have been processed.

The procedure `GenerateNewActiveSets` gives an efficient implementation of generation of new active sets in each iteration of `Collapsing-FLAME`, and takes the currently processed sets $\Delta = \Delta_{(i-1)}$ and a newly processed set $s = s^*_{(i)}$ as input. Let $|s| = k$. In this procedure, $\Delta^k \subseteq \Delta \cup \{s\}$ denotes the set of all processed covariate-sets in Δ of size k, and also includes s. Inclusion of s in Δ^k may lead to generation of new active sets of size $k + 1$ if all of its subsets of size k (one less) have been already processed. The new active sets triggered by inclusion of s in Δ^k would be supersets r of s of size $k + 1$ if all subsets $s' \subset r$ of size $|s'| = k$ belong to Δ^k. To generate such candidate superset r, we can append s with all covariates appearing in some covariate-set in Δ except those in s. However, this naive approach would iterate over many superfluous candidates for active sets. Instead, `GenerateNewActiveSets` safely prunes some such candidates that cannot be valid active sets using **support** of each covariate e in Δ^k, which is the number of sets in Δ^k containing e. Indeed, for any covariate that is not frequent enough in Δ^k, meaning with support less than k, the monotonicity property ensures that any covariate-set that contains that covariate cannot be active. The following proposition shows that this pruning step does not eliminate any valid active set (proof is in the appendix):

Proposition 4.1. *If for a superset r of a newly processed set s where $|s| = k$ and $|r| = k + 1$, all subsets s' of r of size k have been processed (i.e. r is eligible to be active after s is processed), then r is included in the set Z returned by* `GenerateNewActiveSets`.

Algorithm 3: Procedure `GenerateNewActiveSets`	**Example**
Input : s a newly dropped set of size k,	$s = \{2, 3\}$, $k = 2$,
Δ the set of previously processed sets	$\Delta = \{\{1\}, \{2\}, \{3\}, \{5\}, \{1, 2\}, \{1, 3\}, \{1, 5\}\}$
Initialize : `new active sets` $Z = \emptyset$	$Z = \emptyset$
– compute all subsets of Δ of size k and include s	
$\Delta^k = \{\delta \in \Delta \mid size(\delta) = k\} \cup \{s\}$	$\Delta^2 = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 5\}\}$
Notation: $\mathcal{S}_e$ = support of covariate e in Δ^k	$\mathcal{S}_1 = 3, \mathcal{S}_2 = 2, \mathcal{S}_3 = 2, \mathcal{S}_5 = 1$
– get all the covariates contained in sets in Δ^k	
$\Gamma = \{\gamma \mid \gamma \in \delta \text{ and } \delta \in \Delta^k\}$	$\Gamma = \{1, 2, 3, 5\}$
– get the covariates that have enough support minus s	
$\Omega = \{\alpha \mid \alpha \in \Gamma \text{ and } \mathcal{S}_e \geq k\} \setminus s$	$\Omega = \{1, 2, 3\} \setminus \{2, 3\} = \{1\}$
– if all covariates in s have enough support in Δ^k	
if $\{\forall e \in s : \mathcal{S}_e \geq k\}$ **then**	$True$:
– generate new active set	$\alpha = 1$
for *all* $\alpha \in \Omega$ **do**	$r = \{2, 3\} \cup \{1\} = \{1, 2, 3\}$
$r = s \cup \{\alpha\}$	$True$ (subsets of r of size 2 are $\{1, 2\}, \{1, 3\}, \{2, 3\}$)
if *all subsets* $s' \subset r$, $\lvert s' \rvert = k$, *belong to* Δ^k **then**	
– add newly active set r to Z	
add r to Z	$Z = \{\{1, 2, 3\}\}$
return Z	**return** $Z = \{\{1, 2, 3\}\}$

The explicit verification step of whether all possible subsets of r of one less size belongs to Δ^k is necessary, i.e., the above optimization only prunes some candidate sets that are guaranteed not to be active. For instance, consider $s = \{2, 3\}$, $k = 2$, and $\Delta^2 = \{\{1, 2\}, \{1, 3\}, \{3, 5\}, \{5, 6\}\} \cup \{\{2, 3\}\}$. For the superset $r = \{2, 3, 5\}$ of s, all of $2, 3, 5$ have support of ≥ 2 in Δ^2, but this r cannot become active yet, since the subset $\{2, 5\}$ of r does not belong to Δ^2.

To facilitate matching with replacement as in the AEMR objective and to keep track of main and auxiliary matched groups, the algorithm uses a variant of `GroupedExactMatch` given in Algorithm 1 (denoted by `GroupedExactMatch`*). In an iteration i, the `GroupedExactMatch`* procedure takes the entire set of units D, the set of unmatched units from the previous iteration $D_{(i-1)}$, and the covariate-set $J \setminus s^*_{(i)}$ to match on in this iteration. Instead of matching only the unmatched units in $D_{(i-1)}$ using the group-by operator, it matches all units in D to allow for matching with replacement. It keeps track of the main matched groups for the unmatched units $D_{(i-1)}$, and auxiliary groups (see previous section) for the other units. That is, for the units appearing in $D_{(i-1)}$, the returned matched groups are main matched groups since these units are matched for the first time. For all the other units, they may

participate as auxiliary matched units to these same groups. If instead of `GroupedExactMatch`*, the original `GroupedExactMatch` is invoked, the algorithm will perform matching without replacement. The pseudocode for `GroupedExactMatch`* is very similar to `GroupedExactMatch` and is omitted. Finally, the following theorem states the correctness of the `Collapsing-FLAME` algorithm (proof is in the appendix).

Theorem 4.1. *(Correctness) The algorithm* `Collapsing-FLAME` *solves the AEMR problem.*

5 Simulations

We present results under a variety of data generating processes. "generic" FLAME refers to the original FLAME algorithm. We show that Collapsing FLAME produces higher quality matches than popular matching methods such as 1-PNNSM, Nearest Neighbor matching with Mahalanobis distance, and can produce better treatment effect estimates than black box machine learning methods such as Causal Forest (which is not a matching method, and is not interpretable). The 'MatchIt' R-package ([7]) was used to perform 1-Propensity Score Nearest Neighbor matching (1-PSNNM) and Nearest Neighbor with Mahalanobis distance (Mahalanobis). For Causal Forest, we used the 'grf' R-package ([2]). Collapsing FLAME also improves over generic FLAME with regards to the quality of matches in high dimensions. Other matching methods (optmatch, cardinality match) do not scale to large problems and thus needed to be omitted.

Throughout this section, the outcome is generated with

$$y = \sum_i \alpha_i x_i + T \sum_{i=1} \beta_i x_i + T \cdot U \sum_{i,\gamma,\gamma>i} x_i x_\gamma$$

where $T \in \{0, 1\}$ is the binary treatment indicator.

We vary the distribution of the covariates, coefficients (αs, βs, and U), as well as the fraction of treated units.

5.1 Presence of irrelevant covariates

A basic sanity check for matching algorithms is how sensitive to irrelevant covariates they are. To that end, we run experiments with a majority of the covariates being irrelevant to the outcome. For important covariates $1 \leq i \leq 5$, let $\alpha_i \sim N(10s, 1)$ with $s \sim \text{Uniform}\{-1, 1\}$, $\beta_i \sim N(1.5, 0.15)$, $x_i \sim \text{Bernoulli}(0.5)$. For unimportant covariates $5 < i \leq 15$, $x_i \sim \text{Bernoulli}(0.1)$ in the control group and $x_i \sim \text{Bernoulli}(0.9)$ in the treatment group.

Results: As Figure 1 shows, collapsing FLAME and generic FLAME achieve the optimal result before dropping any important covariates. They respectively match 79% and 95% of units on all important covariates. When imposing that the FLAME algorithms find all the matches possible even if important covariates are dropped (not recommended), poor matches are introduced. However, their worst case scenario is still substantially better than the comparative methods, all of which perform poorly in presence of irrelevant covariates. Causal Forest is especially ill suited for this case.

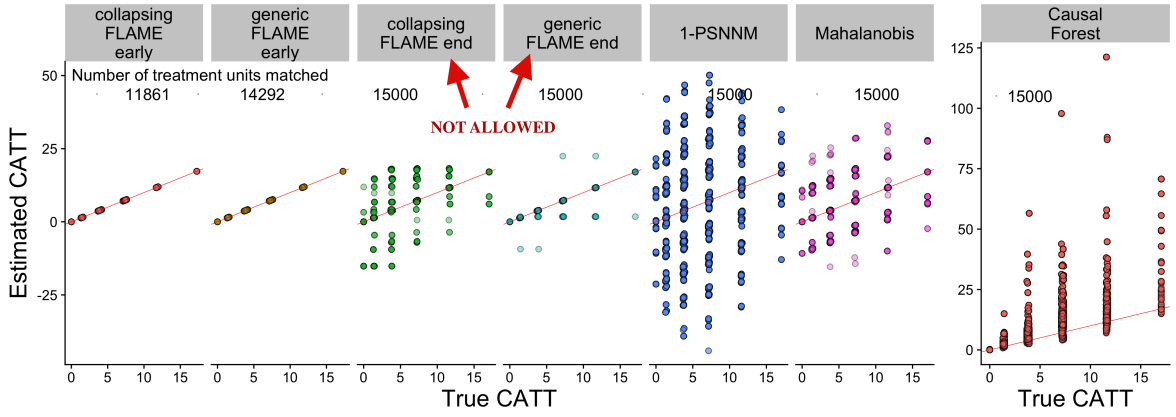


Figure 1: Collapsing FLAME and Generic FLAME perfectly estimate the CATTs when stopped early i.e. before dropping important covariates. Collapsing FLAME and generic FLAME find all the good matches early on and should not be run all the way to the end. If run to the end, poor matches are introduced. All the other methods are sensitive to irrelevant covariates and give poor estimates.

5.2 Exponentially decaying covariate importance

A notable advantage of collapsing FLAME over generic FLAME is that it should produce high quality matches early on. To test that theory, in this experiment we consider covariates of decaying importance (letting the α parameters decrease exponentially). In this simulation, 18 covariates, 15000 treatment units and 15000 control units were used. We let $\alpha_i = 20 \times \left(\frac{4}{5}\right)^i, \beta_i \sim N(1.5, 0.15), x_i \sim$ Bernoulli(0.5). We evaluate the performance of the algorithms when $\approx$ 30%, 50% and 65% of the units are matched.

	Threshold 1		Threshold 2		Threshold 3	
	CollapsingF	GenericF	CollapsingF	GenericF	CollapsingF	GenericF
MSE |	**2.97**	5.02	**4.09**	6.69	**5.23**	8.24

Table 1: MSE when $\approx$ 30%, $\approx$ 50%, and $\approx$ 65% of units are matched for Collapsing FLAME (CollapsingF) and Generic FLAME (GenericF).

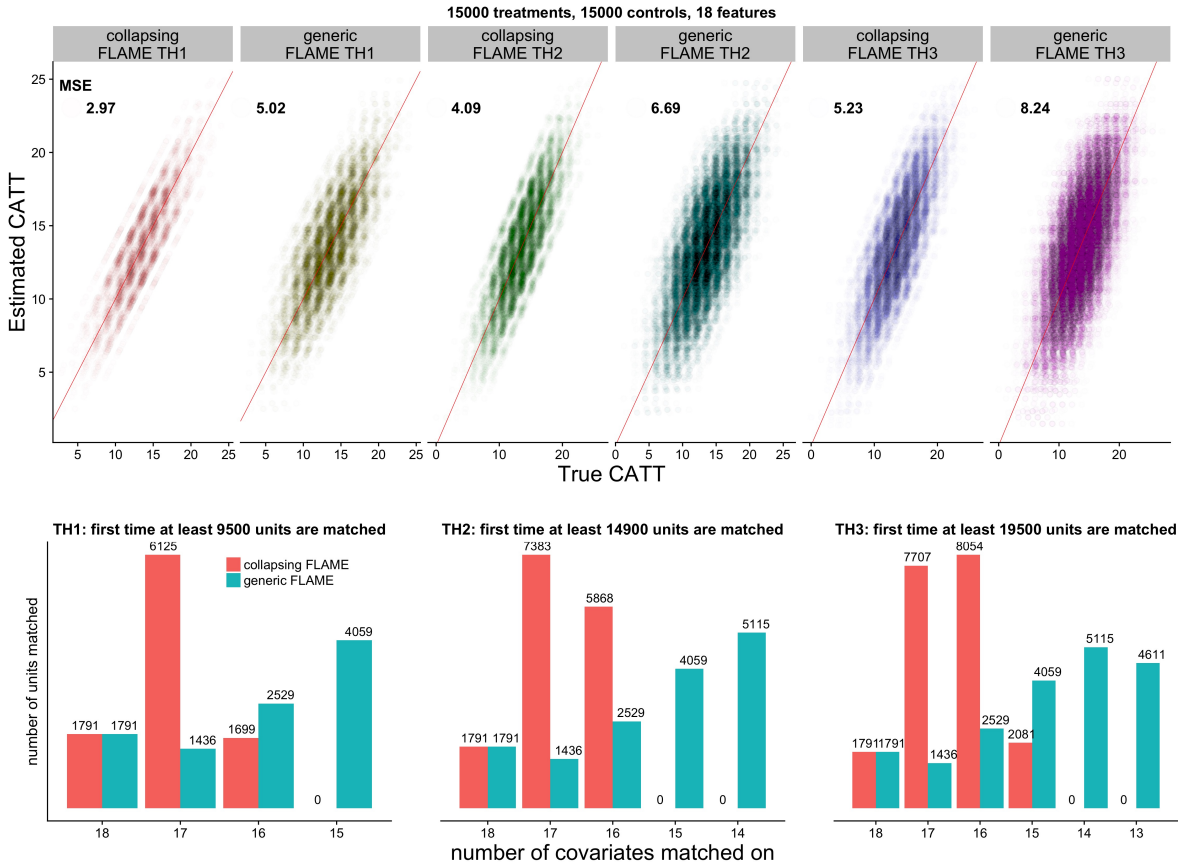


Figure 2: (**Top**): Collapsing FLAME makes higher quality matches early on and outperforms generic FLAME on all thresholds. (**Bottom**): collapsing FLAME consistently matches on more covariates than generic FLAME and minimizes the Hamming distance between matched units.

Results: As shown in Figure 2 and Table 1, collapsing FLAME produces higher quality matches early on. In the first threshold where $\approx 30\%$ of the units are matched, collapsing FLAME provides good estimation of the CATTs. As more units are matched ($\approx 50\%$ then $\approx 65\%$), collapsing FLAME maintains good CATT estimate performance, whereas generic FLAME produces poorer estimates. Moreover, collapsing FLAME consistently matches units on more covariates than generic FLAME, thus, better achieving the goals of *almost-exact matching*.

5.3 Imbalanced data simulation

A common situation in observational studies is the presence of imbalance in the data: that is there are substantially more control than treatment units. Imbalanced data provides a unique opportunity for matching methods to produce the best possible matches for treated units. The model considered in this experiment consists of covariates of decreasing importance and does not include unimportant

covariates: $x_i \sim \text{Bernoulli}(0.5)$, $\alpha_i = 20 \times \left(\frac{4}{5}\right)^i$, and $\beta \sim \text{Normal}(1.5, 0.15)$. A fixed batch of 2000 treatment units was generated as well as a batch of 40000 control units. To analyse the impact of the imbalance on the different matching procedures, we sample from the batch of controls to construct different imbalance ratios: 40000 in the most imbalanced case, then 20000 and, finally, 10000. This experiment was run reusing control units to find the best possible match for each treatment unit.

Results: In terms of covariates matched on, collapsing FLAME yields better results when the data is extremely imbalanced (*Figure 3*). Collapsing FLAME outperforms generic FLAME in these settings as it is able to match more units on substantially more covariates the more imbalanced the data are. Because control units are reused, the high dimensional matching of collapsing FLAME is amplified. In this experiment, collapsing FLAME has an average of 4 covariates not matched on, with $\approx 84\%$ of units matched on everything but 2 covariates. On the other hand, generic FLAME averages 7 covariates not matched on and only $\approx 25\%$ units matched on everything but 2 covariates.

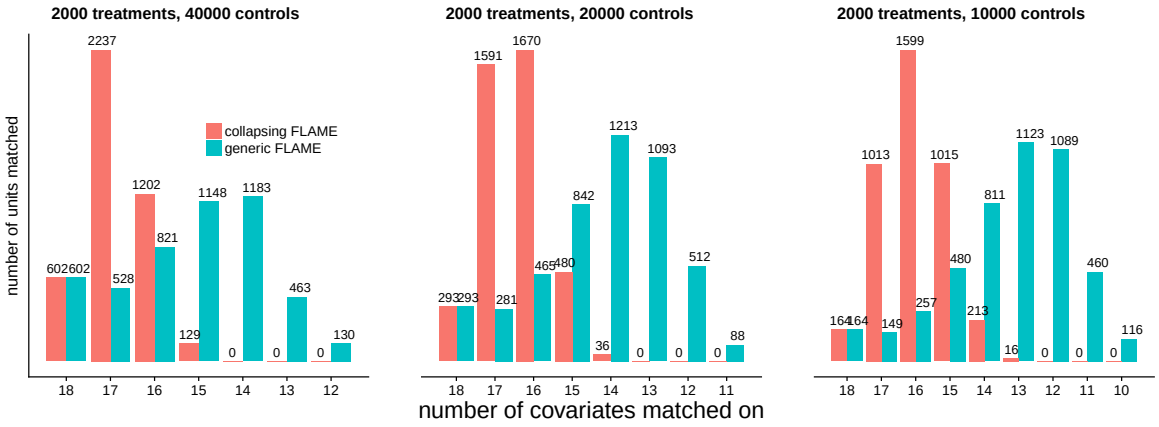


Figure 3: For all the different imbalance ratios, collapsing FLAME consistently matches on more covariates than generic FLAME.

	Mean Squared Error (MSE)		
	Ratio 1 (more imbalance)	Ratio 2	Ratio 3 (less imbalance)
Collapsing FLAME	**3.83**	**5.62**	**8.17**
Generic FLAME	7.45	10.79	13.92
Mahalanobis	34.59	41.56	63.40
1-PSNNM	296.32	337.26	335.23

Table 2: MSE for different imbalance ratios: ratio 1 = 2000 treatments, 40000 controls, ratio 2 = 2000 treatments, 20000 controls, ratio 3 = 2000 treatments, 10000 controls

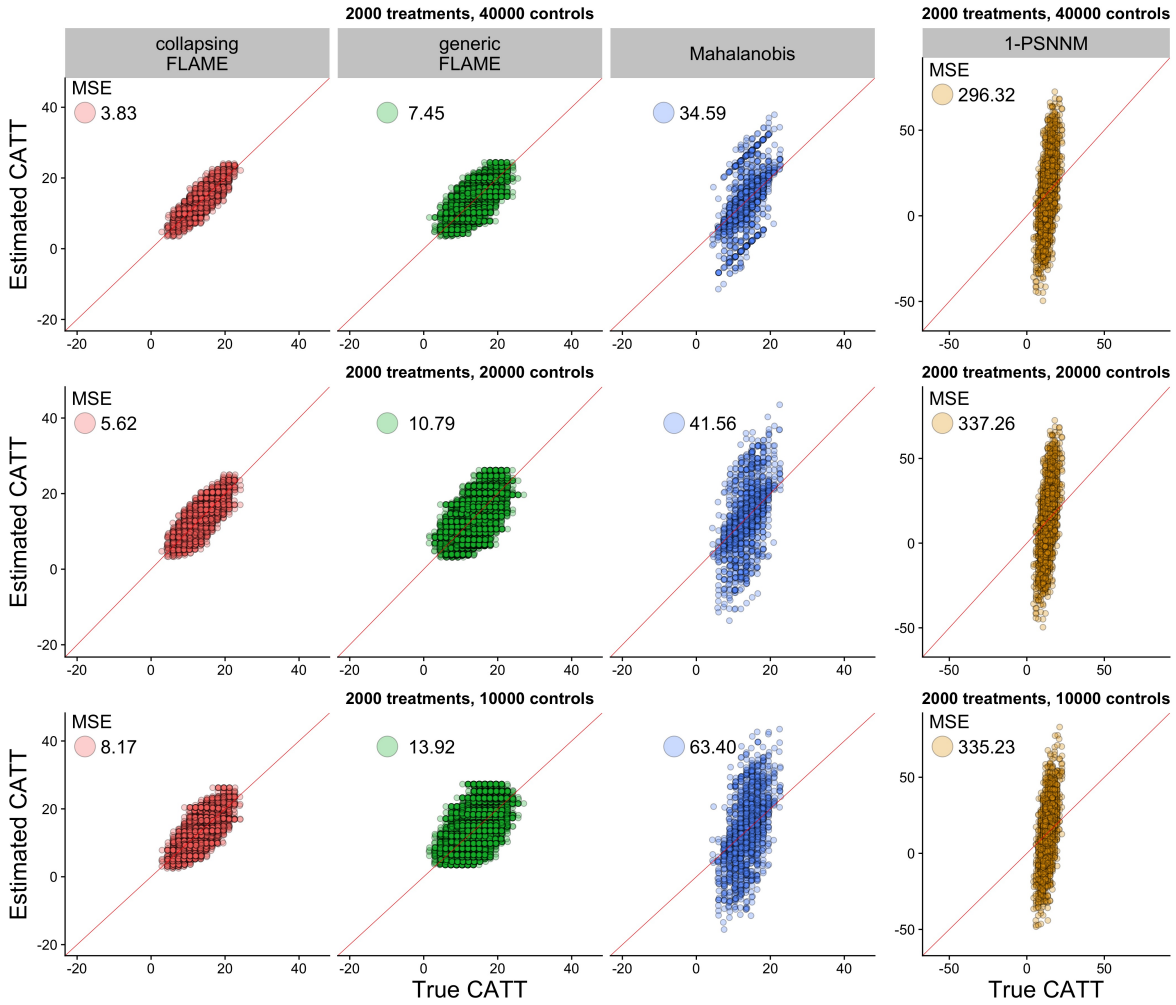


Figure 4: (1) Estimated CATT vs True CATT from less imbalance (bottom) to more imbalance (top). In all cases, the FLAME algorithms outperforms 1-PSNNM and Mahalanobis, which produce poor quality estimates. (2) collapsing FLAME estimation quality increases with the imbalance. The best results are produced when the data is extremely imbalanced. Within each imbalance ratio, collapsing FLAME outperforms generic FLAME.

(*Figure 4*) While the performance of non-FLAME competitors does not depend on the fraction of treated and control units, the FLAME algorithms thrive when the number of potential control units to match to grows.A first check reveals that the FLAME algorithms outperform the neareast neighbor matching methods, which produce poor estimates unrelated to the degree of imbalance (see MSE). A second check on the FLAME algorithms highlights that collapsing FLAME is distinctively better than generic FLAME when the data are imbalanced.

5.4 Missing Data

Here we demonstrate the performance of collapsing FLAME when data are missing at random. We compared performance between generic and collapsing FLAME both with and without multiple imputations. To generate covariates X we first sample $Z \sim N_p(\mu, \Sigma)$, where Σ is not the identity matrix and then let $X_j = f_j(Z_j)$ where $f_j(z) = 1_{z>0}$. This creates correlated binary variables so methods that impute the missing values are expected to perform well. We generate 15000 control and 5000 treated units; 20% of the data are missing at random.

To allow for missing values in the FLAME algorithms we construct an $n \times p$ matrix O where $o_{ij} = 1$ if covariate j is unobserved for unit i. Treating "missing" as just another category for each variable we proceed with the algorithm by adding a condition for a matched group to be valid: if covariates $J' = \{j_1, \ldots, j_{p'}\}$ are being matched on then $\sum_{j \in J'} o_{ij} = 0$ for a unit i in the group. If the sum is greater than 0 then the group matched on the level "missing" for at least one covariate, making it invalid. We compare this approach to multiply imputing 10 datasets using the Multiple Imputation Chained Equations algorithm in the 'mice' R package and averaging the estimates over the datasets. The correlation matrix Σ that defines the relationship among the covariates X is given in Figure 5.

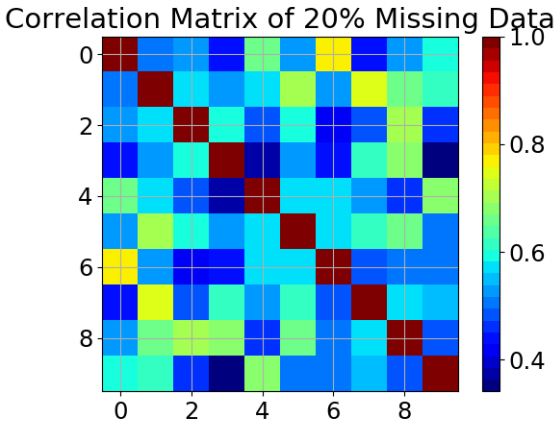
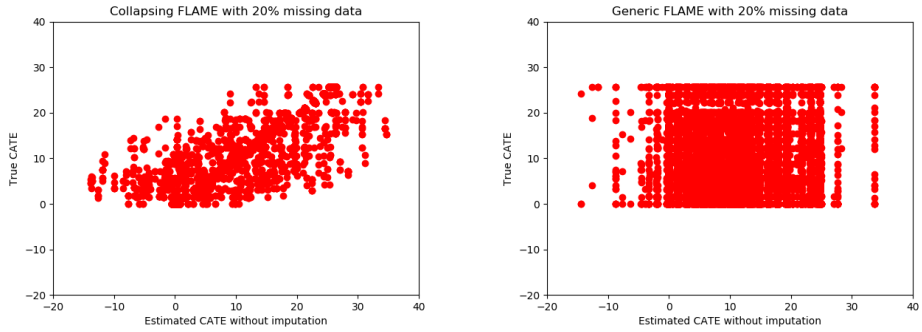


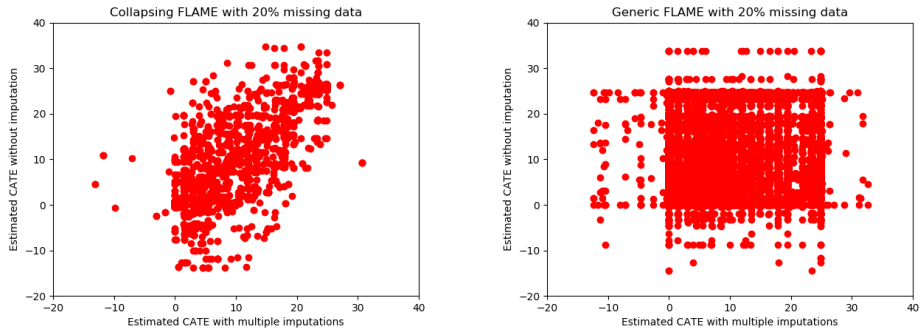
Figure 5: Correlation matrix that defines the relationship among the covariates X.

Results: Collapsing FLAME without imputation outperforms generic FLAME without imputation and nearly matches FLAME with imputations in this setting, both in terms of mean squared error (5.08 vs 195.6 vs 2.77) and qualitatively in terms of CATE estimation. By multiple imputing and using collapsing FLAME we achieve a small reduction in MSE to 1.11. This suggests that for datasets that are too large to undergo multiple imputation collapsing FLAME still produces reasonable causal estimates.



(a) collapsing FLAME (b) generic FLAME

Figure 6: Comparison of True CATE and estimated CATE without imputations for Collapsing FLAME and Generic FLAME. 20% missing values



(a) collapsing FLAME (b) generic FLAME

Figure 7: Comparison of CATE estimation with missing data and using multiple imputation for Collapsing FLAME and Generic FLAME. 20% missing values

5.5 Running Time evaluation

In this section, we compare the running time of our dynamic approach to solving the AEMR problem with a brute force solution. We also compare with generic FLAME which does not solve the AEMR problem. Figure 8 shows how the running time of these different methods varies with the number of units n and the number of covariates p. All experiments were run on a MacBook Pro with Intel Core i5 Processor (Cores: 2, Speed: 2,4 GHz), 8 GB RAM.

Results: Generic FLAME provides the best running time performance because it utilizes a bit vector implementation and incrementally decreases the covariate space. On the other hand, collapsing FLAME resets the pool of covariate-sets to consider at each round of the algorithm. However, as shown in the previous simulations, collapsing FLAME produces high quality matches that the other methods do not. Our proposed method substantially reduces the running time for solving the AEMR problem

as compared to the brute force approach. The running time for collapsing FLAME can be further optimized through simple parallelization of the checking of active sets.

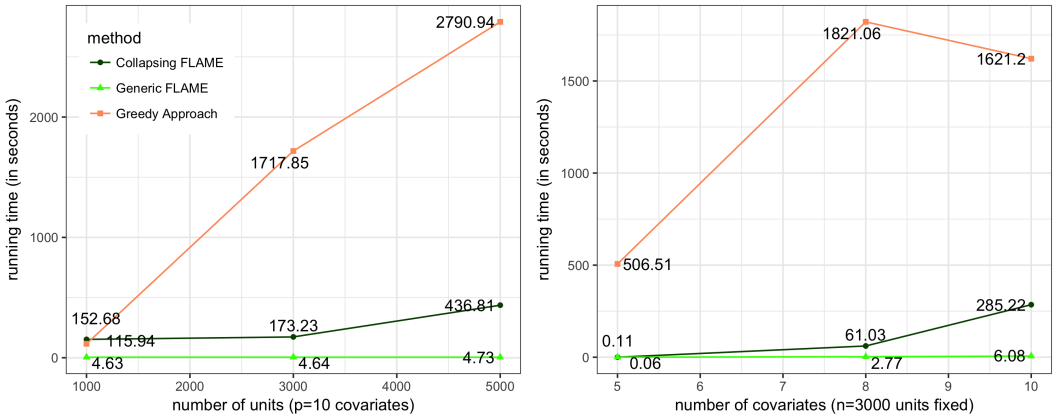


Figure 8: Run time comparison.

6 Breaking the Cycle of Drugs and Crime in the United States

Breaking The Cycle (BTC) [6] is a social program conducted in several U.S. states designed to reduce criminal involvement and substance abuse, and improve health and employment among current offenders. A survey [6] was conducted in Alabama, Florida, and Washington regarding the program's effectiveness, with high quality data for over 380 individuals. These data (and this type of data generally) can be a powerful tool in the war against opiods, and our ability to draw interpretable, trustworthy conclusions from it depends on our ability to construct high-quality matches. For the survey, participants were chosen to receive screening shortly after arrest and participate in a drug intervention under supervision. Similar defendants before the start of the BTC program were selected as the control group.

The features used for the BTC data are listed in Table 3. Each feature has a variable importance score associated with it, the ratio of the loss of a ridge regression model trained with and without randomly permuting that feature. To obtain the values within the table, we randomly shuffled the values of that feature 100 times, retraining ridge regression, and each time compared with the unshuffled loss of ridge regression: $\frac{\text{average}(\widehat{\text{loss}}_{\text{removing } i})}{\text{loss}}$.

In Table 3, the mean variable importance values are reported. These calculations were performed on a holdout training set, not on the test data.

As Table 3 shows, all covariates have similar values of the importance score, which suggests that all covariates are almost equally important.

Feature	Importance Score
Live with anyone with an alcohol problem	0.958458
Have trouble understanding in life	0.983272
Live with anyone using nonprescribed drugs	0.984558
Have problem getting along with father in life	0.989829
Have a automobile	0.990346
Have driver license	0.990732
Have serious depression or anxiety in past 30 days	0.992095
Have serious anxiety in life	0.994400
SSI benefit Last 6 Months	1.001295
Have serious depression in life	1.003302

Table 3: Features for BTC data, and importance score of each feature, learned from a holdout training set.

Order	Basic FLAME	Collapsing FLAME
1st	Live with anyone with an alcohol problem	Live with anyone with an alcohol problem
2nd	Have serious anxiety in life	Have trouble understanding in life
3rd	Live with anyone using nonprescribed drugs	Have serious depression or anxiety in past 30 days
4th	Have trouble understanding in life	Live with anyone using nonprescribed drugs
5th	Have problem getting along with father in life	Have serious anxiety in life
6th	Have serious depression in life	Have driver license
7th	Have driver license	Have problem getting along with father in life
8th	Have a automobile	Have serious anxiety in life
9th	Have serious depression or anxiety in past 30 days	Have a automobile

Table 4: Order in which features were processed for basic FLAME and collapsing FLAME.

6.1 Order of Dropping Covariates

The order in which collapsing FLAME and basic FLAME process covariates is different. Table 4 shows the order in which the covariates were processed for both algorithms. We used the formulations of

FLAME and collapsing FLAME where variable importance is recomputed for only the variables we are considering at that iteration, since variable importance values change in the absence of various covariates. Covariates in FLAME are not eliminated on variable importance alone, there is a balancing factor that prefers not to eliminate too many units from either treated or control at once. Because collapsing FLAME eliminates one variable at a time, it tends to process the least relevant covariate in the earlier rounds. For example, at the first round, both algorithms drop the covariate"Live with anyone with an alcohol problem" which has the lowest importance score in Table 3. At the second round, collapsing FLAME process the covariate "Have trouble understanding in life" because this covariate has the lowest importance score aside from "Live with anyone with an alcohol problem". On the other hand, at that same second round, basic FLAME processes "Have serious anxiety in life" which now is dropped along with "Live with anyone with an alcohol problem" but does not have the least importance score.

6.2 Number of units matched

We compare the quality of matches between basic FLAME and Collapsing FLAME, in terms of the number of covariates used to match within the groups. Many of the units matched exactly on all covariates and thus were matched by both algorithms at the first round. For the remaining units, collapsing FLAME matches on more covariates than basic FLAME. Figure 9 shows the results. In particular, collapsing FLAME matched many more units on 9 variables, whereas basic FLAME matched more units on only one variable.

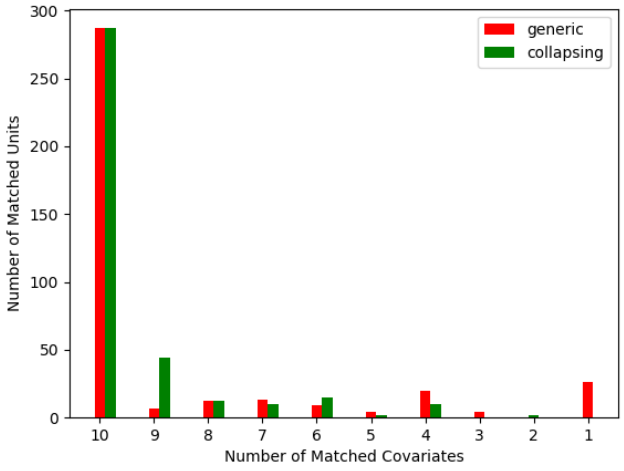


Figure 9: Comparison of the number of covariates matched on for the matched units

6.3 Comparison with Minimax Surrogate Loss Approach

We wanted to use collapsing FLAME as a tool to double check performance of a black box machine learning approach. There is no ground truth, so there will not be formal accuracy evaluation (see simulations for accuracy comparison). We chose a recent method that predicts whether treatment effects are positive, negative, or neutral, using a support vector machine formulation [21].

We ran collapsing FLAME on the BTC dataset and saved the CATE for each treatment and control unit that were matched. Units with a positive CATE have a negative treatment effect and vice versa. We also implemented the SVM approach and recorded a prediction of positive, negative, or neutral treatment effect for each unit. As Figure 10 shows, collapsing FLAME and the SVM approach agree on most of the matched units.

To see this, the units for which collapsing FLAME predicted approximately zero treatment effect all have a "neutral" treatment effect label from the SVM approach.

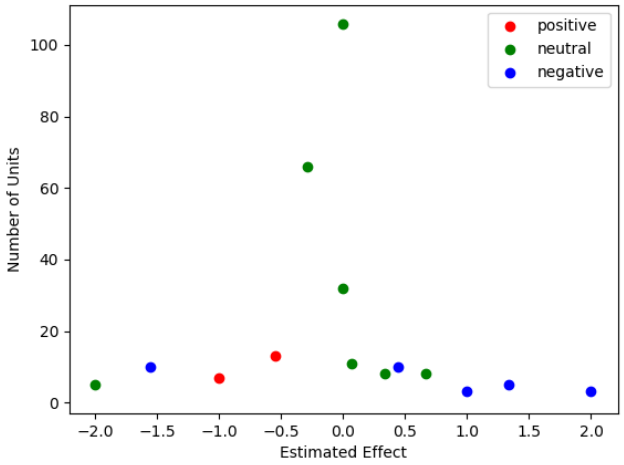


Figure 10: CATE estimates by collapsing FLAME and treament effect classification by SVM approach

Most positive CATE's corresponded to negative treatment effects from the SVM. Only one matched group seems to have a different prediction than the SVM: a negative CATE with a negative treatment effect prediction. The easiest way to explain the discrepancy between the two methods is that collapsing FLAME is a matching method, not a statistical model. Usually if one wanted to estimate CATEs, one would smooth the treatment effect estimates with a regression model after matching, however, we did not do this. Thus, if the negative SVM group happened to be closer in proximity to the matched groups with positive CATEs, it would completely resolve the discrepancy between the two models: the SVM simply smoothed out the treatment effect estimates so that there was a negative predicted treatment effect, even though the CATE may have been negative for that matched group.

In order to determine whether this was true, we computed the Hamming distance between the units in that unusual group to units in other groups to investigate. As it turned out, the units within that matched group were much closer to the units with negative treatment effect than other units, and as such, the difference between the results of the two methods is because there was no smoothing in the matching method. We similarly investigated the blue group at x=0.5 CATE estimate and again, the covariates of its units were closer in Hamming distance to other blue groups than to other points.

Smoothing the matched groups is beyond the scope of this paper, but can be done with any regression method once the groups are formed if desired. The combination of matched groups with regression techniques for treatment effect can easily handle data with highly nonlinear baseline effects and smooth treatment effects; matching methods allow us to create such models without having to model the nonlinear baseline at all since the matching accounts for it.

7 Conclusion

The method introduced here produces the highest possible quality matches for matching in categorical datasets. Despite the problem's inherent hardness, the algorithm is practical for real problems, owing to the use of fast database queries and a monotonicity property. Code is publicly available at https://github.com/almostExactMatch/collapsingFLAME.

References

[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[2] S. Athey, J. Tibshirani, and S. Wager. Generalized Random Forests. *ArXiv e-prints*, October 2016.

[3] Alexandre Belloni, Victor Chernozhukov, and Christian Hansen. Inference on treatment effects after selection among high-dimensional controls. *The Review of Economic Studies*, 81(2):608–650, 2014.

[4] William G Cochran and Donald B Rubin. Controlling bias in observational studies: A review. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 417–446, 1973.

[5] Max H Farrell. Robust inference on average treatment effects with possibly more covariates than observations. *Journal of Econometrics*, 189(1):1–23, 2015.

[6] Adele V. Harrell, Douglas Marlowe, and Jeffrey Merrill. Breaking the cycle of drugs and crime in birmingham, alabama, jacksonville, florida, and tacoma, washington, 1997-2001. 2006.

[7] Daniel Ho, Kosuke Imai, Gary King, and Elizabeth Stuart. Matchit: Nonparametric preprocessing for parametric causal inference. *Journal of Statistical Software, Articles*, 42(8):1–28, 2011.

[8] Stefano M Iacus, Gary King, and Giuseppe Porro. Causal inference without balance checking: Coarsened exact matching. *Political analysis*, page mpr013, 2011.

[9] Stefano M Iacus, Gary King, and Giuseppe Porro. Multivariate matching methods that are monotonic imbalance bounding. *Journal of the American Statistical Association*, 106(493):345–361, 2011.

[10] Jeremy A Rassen and Sebastian Schneeweiss. Using high-dimensional propensity scores to automate confounding control in a distributed medical product safety surveillance system. *Pharmacoepidemiology and drug safety*, 21(S1):41–49, 2012.

[11] Paul R Rosenbaum. Imposing minimax and quantile constraints on optimal matching in observational studies. *Journal of Computational and Graphical Statistics*, (just-accepted), 2016.

[12] Paul R Rosenbaum and Donald B Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

[13] S. Roy, C. Rudin, A. Volfovsky, and T. Wang. FLAME: A Fast Large-scale Almost Matching Exactly Approach to Causal Inference. *ArXiv e-prints*, July 2017.

[14] Donald B Rubin. Matching to remove bias in observational studies. *Biometrics*, pages 159–183, 1973.

[15] Donald B Rubin. The use of matched sampling and regression adjustment to remove bias in observational studies. *Biometrics*, pages 185–203, 1973.

[16] Donald B Rubin. Multivariate matching methods that are equal percent bias reducing, i: Some examples. *Biometrics*, pages 109–120, 1976.

[17] Donald B. Rubin. Randomization analysis of experimental data: The fisher randomization test comment. *Journal of the American Statistical Association*, 75(371):591–593, 1980.

[18] Donald B Rubin. For objective causal inference, design trumps analysis. *The Annals of Applied Statistics*, pages 808–840, 2008.

[19] Babak Salimi, Corey Cole, Dan R. K. Ports, and Dan Suciu. Zaliql: Causal inference from observational data at scale. *PVLDB*, 10(12):1957–1960, 2017.

[20] Sebastian Schneeweiss, Jeremy A Rassen, Robert J Glynn, Jerry Avorn, Helen Mogun, and M Alan Brookhart. High-dimensional propensity score adjustment in studies of treatment effects using health care claims data. *Epidemiology (Cambridge, Mass.)*, 20(4):512, 2009.

[21] S. Thye Goh and C. Rudin. A Minimax Surrogate Loss Approach to Conditional Difference Estimation. *ArXiv e-prints*, March 2018.

[22] Stefan Wager and Susan Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, (just-accepted), 2017.

A Omitted Proofs

A.1 Proof of Proposition 4.1

Proposition 4.1 *If for a superset r of a newly processed set s where $|s| = k$ and $|r| = k + 1$, all subsets s' of r of size k have been processed (i.e. r is eligible to be active after s is processed), then r is included in the set Z returned by* `GenerateNewActiveSets`.

Proof. Suppose all subsets of r of size k are already processed and belong to Δ^k. Let f be the covariate in $r \setminus s$. Clearly, f would appear in Δ^k, since at least one subset $s' \neq s$ of r of size k would contain f, and $s' \in \Delta^k$. Further all covariates in r, including f and those in s will have support at least k in Δ^k. To see this, note that there are $k + 1$ subsets of r of size k, and each covariate in r appears in exactly k of them. Hence $f \in \Omega$, the 'if' condition to check minimum support for all covariates in s is satisfied, and the final 'if' condition to eliminate false positives is satisfied too. Therefore r will be included in Z returned by the procedure. □

A.2 Proof of Theorem 4.1

Theorem 4.1*(Correctness)* *The algorithm* `Collapsing-FLAME` *solves the AEMR problem.*

Proof. Consider any treatment unit t. Let s be the set of covariates in its main matched group returned in `Collapsing-FLAME` (the while loop in `Collapsing-FLAME` runs as long as there is a treated unit, and the `GroupedExactMatch`* returns main matched group for every unit when it is matched for the first time). Let $\mathbf{v}_s$ be the indicator vector of s (see (1)). Since the `GroupedExactMatch`* procedure returns a main matched group only if it is a *valid* matched group containing at least one treated and one control unit (see Algorithm 1), and since all units in the matched group on s have the same value of covariates in $J \setminus s$, there exists a unit ℓ with $T(\ell) = 0$ and $\mathbf{x}_\ell \circ \mathbf{v}_s = \mathbf{x}_t \circ \mathbf{v}_s$.

Hence it remains to show that the covariate set s in the main matched group for t corresponds to the maximum weight $\mathbf{v}^T\mathbf{w}$. Assume the contradiction that there exists another covariate-set r such that $\mathbf{v}_r^T w > \mathbf{v}_s^T w$, there exists a unit ℓ' with $T(\ell') = 0$ and $\mathbf{x}_{\ell'} \circ \mathbf{v}_r = \mathbf{x}_t \circ \mathbf{v}_r$, and gives the maximum weight $\mathbf{v}_r^T w$ over all such r.

(i) r cannot be a (strict) subset of s, since `Collapsing-FLAME` ensures that all subsets are processed before a superset is processed to satisfy the downward closure property in Proposition 3.1.

(ii) r cannot be a (strict) superset of s, since it would violate the assumption that $\mathbf{v}_r^T w > \mathbf{v}_s^T w$ for non-negative weights.

(iii) Hence assume that r and s are incomparable (there exist covariates in both $r \setminus s$ and $s \setminus r$). Suppose the active set s was chosen in iteration i. If r was processed in an earlier iteration $j < i$, since r forms a valid matched group for t, it would give the main matched group for t violating the assumption. We argue that in this case r must be active at the start of iteration i, and will be chosen as the best covariate set in iteration i, leading to a contradiction.

Note that we start with all singleton sets as active sets in $\Delta_{(0)} = \{\{1\}, \cdots, \{p\}\}$ in the `Collapsing-FLAME` algorithm. Consider any singleton subset $r_0 \subseteq r$ (comprising a single covariate in r). Due to the downward closure property in Proposition 3.1, $\mathbf{v}_{r_0}^T w \geq \mathbf{v}_r^T w$, hence $\mathbf{v}_{r_0}^T w \geq \mathbf{v}_r^T w > \mathbf{v}_s^T w$. Hence all of these singleton subsets of r will be processed in earlier iterations $j < i$, and will belong to the set of processed covariate sets $\Lambda_{(i-1)}$.

Repeating the above argument, consider any subset $r' \subseteq r$. It holds that $\mathbf{v}_{r'}^T w \geq \mathbf{v}_r^T w > \mathbf{v}_s^T w$. Hence all subsets r' of r will be processed in earlier iterations $j < i$ starting with the singleton subsets of r. In particular, all subsets of size $|r| - 1$ will belong to $\Lambda_{(i-1)}$. As soon as the last of those subsets is processed, the procedure `GenerateNewActiveSets` will include r in the set of active sets in a previous iteration $j < i$. Hence if r is not processed in an earlier iteration, it must be active at the start of iteration i, leading to a contradiction.

Hence for all treatment units t, the covariate-set r giving the maximum value of $\mathbf{v}_r^T w$ will be used to form the main matched group of t, showing the correctness of the `Collapsing-FLAME` algorithm. □

B Adjustment of weights

One noticeable feature of `Collapsing-FLAME` resides in it ability to dynamically adjust the weight of the covariates througout the run of the algorithm. This property is important as covariates become more or less relevant in presence of other covariates. To recompute the weight ie the importance score of each covariate, `Collapsing-FLAME` uses a hold-out training set D^H whose covariate space is restricted to the subset of the covariates being considered ($\mathcal{J} \setminus s$). Using the hold-out set $D^H = [X^H, Y^H, T^H]$ that includes both treatment and control units, we obtain the vectors representing the weights β_t and β_c by linear regressions on control and treatment units. The readjusted weights are used to compute a *prediction error* $\mathcal{PE}$ measure to determine which subset of covariates leads to the best estimation

quality. Specifically, $\mathcal{PE}$ is computed for each active set considered by

$$\begin{aligned} \texttt{PE}(D^H_{\Theta,s}) \quad = \quad & \min_{\beta_t,\beta_c} \Bigg[\frac{1}{\sum(1-T^H_u)} \sum_{u:T^H_u=0} \left(Y^H_u - X^H\,(u, J\ s)\,\beta_c \right)^2 + \\ & \frac{1}{\sum T^H_u} \sum_{u:T^H_u=1} \left(Y^H_u - X^H\,(u, J\ s)\beta_t)^2 \right) \Bigg]. \end{aligned} \tag{2}$$

Notation: $\mathbf{D^H}$ is the holdout dataset, Θ represents a selector function on the rows and columns of the dataset for any given matched unit u. In other words, $D^H_{\Theta,u}$ consists of a subset of rows and covariates that match with u (maximizing Θ gives the main matched group for any unit u), T^H_u and Y^H_u are respectively the treatment indicator and the outcome for unit u in the hold-out dataset.

Other criteria for optimization in addition to optimizing over $\mathbf{v}^T\mathbf{w}$ is giving preference to subsets of attributes that allow for better balancing of treatment and control units in the matched groups. That balancing factor $\mathcal{BF}$ depends only on the number of units matched at each iteration by dropping s and the remaining unmatched units.

$$\mathcal{BF}_{\Theta,s} = \frac{\text{\#of matched controls by dropping } s}{\text{\#of remaining controls}} + \frac{\text{\#of matched treatments by dropping } s}{\text{\#of remainging treatments}} \tag{3}$$

A matched quality quantity consisting of a trade off between these two measures ensures that `Collapsing-FLAME` only processes sets that maintain good prediction quality while the matched groups contain enough units to yield good estimate of treatment effects.

C Database queries: `group-by`

`group-by` is a powerful database query available on most relational database management systems. Distinctive features of database queries are (1) their scalabilty to large dataset that can not fit in memory, (2) a high level syntax that allow for an intuitive call to complex operations. For example, finding covariates units matched on can be achieved in one line (see below). Using database implementations to scale causal inference methods is an emerging field of study (see [19]).

In `GroupedExactMatch` the exact matching step can succintly be computed using `group-by`:

```
WITH tempgroups AS
    (SELECT A_1, A_2, ..., A_p      (matched groups are identified by their covariate values)
    FROM D
    WHERE is_matched = 0      (use data that are not yet matched)
    GROUP BY A_1, A_2, ..., A_p      (create matched groups with identical covariates)
    HAVING SUM(T) >= 1 AND SUM(T) <= COUNT(*)-1      (groups have≥1 treated, but not all treated)
    )
```

Notation: A new column is added to the input dataset to identify units that have been matched. Each time a unit is matched, the value of `is_matched` for that unit is set to 1. Conversely, the unmatched units have a `is_matched` set to 0. $A_1, \cdots, A_p$ are the covariates in $\mathcal{J}$. The *SELECT, WHERE, GROUP-BY clauses* loop through the covariates of the input dataset D (SELECT), identify units that have not been matched yet (WHERE) and finds the covariates on which those units are matched. The `pruning` step (HAVING clause) ensures that the constraint of at least one treatment and one control unit in the matched groups be satisfied. *tempgroups* temporarily contains all groups found.

Once the matched groups are found, the `is_matched` indicator for the newly matched units from tempgroups is updated in the input dataset D.

```
SET is_matched = ℓ
WHERE EXISTS
    (SELECT D.A_1, D.A_2, ..., D.A_p
    FROM tempgroups S      (set of covariate values for valid groups)
    WHERE S.A_1 = D.A_1 AND S.A_2 = D.A_2 AND ··· AND S.A_p = D.A_p
    )
    AND is_matched = 0
```