# Fast Resampling Weighted v-Statistics

**Authored by:**

Chunxiao Zhou
Jiseong Park
Yun Fu

**Abstract**

In this paper, a novel, computationally fast, and alternative algorithm for com- puting weighted v-statistics in resampling both univariate and multivariate data is proposed. To avoid any real resampling, we have linked this problem with finite group action and converted it into a problem of orbit enumeration. For further computational cost reduction, an efficient method is developed to list all orbits by their symmetry order and calculate all index function orbit sums and data function orbit sums recursively. The computational complexity analysis shows reduction in the computational cost from n! or nn level to low-order polynomial level.

## 1 Paper Body

Resampling methods (e.g., bootstrap, cross-validation, and permutation) [3,5] are becoming increasingly popular in statistical analysis due to their high flexibility and accuracy. They have been successfully integrated into most research topics in machine learning, such as feature selection, dimension reduction, supervised learning, unsupervised learning, reinforcement learning, and active learning [2, 3, 4, 7, 9, 11, 12, 13, 20]. The key idea of resampling is to generate the empirical distribution of a test statistic by resampling with or without replacement from the original observations. Then further statistical inference can be conducted based on the empirical distribution, i.e., resampling distribution. One of the most important problems in resampling is calculating resampling statistics, i.e., the expected values of test statistics under the resampling distribution, because resampling statistics are compact representatives of the resampling distribution. In addition, a resampling distribution may be approximated by a parametric model with some resampling statistics, for example, the first several moments of a resampling distribution [5, 16]. In this paper, we focus on computing resampling weighted vstatistics [18] (see Section 2 for the formal definition). Suppose our data includes n observations, a weighted v-statistic is a summation of products of data function terms and index function terms, i.e., weights, over all possible k observations chosen from n observations, where

k is the order of the weighted v-statistic. If we treat our data as points in a multi-dimensional space, a weighted v-statistic can be considered as an average of all possible weighted k-points distances. The higher k, the more complicated interactions among observations can be modeled in the weighted v-statistic. Machine learning researchers have already used weighted v-statistics in hypothesis testing, density estimation, dependence measurement, data pre-processing, and classification [6, 14, 19, 21] . Traditionally, estimation of resampling statistics is solved by random sampling since exhaustive examination of the resampling space is usually ill advised [5,16]. There is a tradeoff between accuracy and computational cost with random sampling. To date, there is no systematic and efficient solution to the issue of exact calculation of resampling statistics. Recently, Zhou et.al. [21] proposed a recursive method to derive moments of permutation distributions (i.e., empirical distribution generated by resampling without replacement). The key strategy is to divide the whole index set (i.e., indices of all possible k observations ) into several permutation equivalent index subsets such that the summa1

tion of the data/index function term over all permutations is invariant within each subset and can be calculated without conducting any permutation. Therefore, moments are obtained by summing up several subtotals. However, methods for listing all permutation equivalent index subsets and calculating of the respective cardinalities were not emphasized in the previous publication [21]. There is also no systematic way to obtain coefficients in the recursive relationship. Even only for calculating the first four moments of a second order resampling weighted v statistic, hundreds of index subsets and thousands of coefficients have to be derived manually. The manual derivation is very tedious and error-prone. In addition, Zhou?s work is limited to permutation (resampling without replacement) and is not applicable to bootstrapping (resampling with replacement) statistics. In this paper, we propose a novel and computationally fast algorithm for computing weighted vstatistics in resampling both univariate and multivariate data. In the proposed algorithm, the calculation of weighted v-statistics is considered as a summation of products of data function terms and index function terms over a high-dimensional index set and all possible resamplings with or without replacement. To avoid any resampling, we link this problem with finite group actions and convert it into a problem of orbit enumeration [10]. For further computational cost reduction, an efficient method has been developed to list all orbits by their symmetry order and to calculate all index function orbit sums and data function orbit sums recursively. With computational complexity analysis, we have reduced the computational cost from n! or nn level to low-order polynomial level. Detailed proofs have been included in the supplementary material. In comparison with previous work [21], this study gives a theoretical justification of the permutation equivalence partition idea and extends it to other types of resamplings. We have built up a solid theoretical framework that explains the symmetry of resampling statistics using a product of several symmetric groups. In addition, by associating this problem with finite group action, we have developed an algorithm to enumerate all orbits by their symmetry order and generated a recursive relationship for orbits

sum calculation systematically. This is a critical improvement which makes the whole method fully programmable and frees ourselves from onerous derivations in [21].

2

Basic idea

In general, people prefer choosing statistics which have some symmetric properties. All resampling strategies, such as permutation and bootstrap, are also more or less symmetric. These facts motivated us to reduce the computational cost by using abstract algebra. Pn This study is focused on computing resampling weighted v-statistics, i.e., T (x) = i1 =1 ? ? ? Pn T w(i , ? ? ? , i )h(x , ? ? ? x ), where x = (x , x , ? ? ? , x ) is a collection of n observa1 d i i 1 2 n 1 d id =1 tions (univariate/multivariate), w is an index function of d indices, and h is a data function of d observations. Both w and h are symmetric, i.e., invariant under permutations of the order of variables. Weighted v-statistics cover a large amount of popular statistics. For example, in the case of multiple comparisons, observations are collected from g groups: first group (x1 , ? ? ? , xn1 ), second group (xn1 +1 , ? ? ? , xn1 +n2 ), and last group (xn ng +1 , ? ? ? , xn ), where n1 , n2 , ? ? ? , ng are numbers of observations in each group. In order to test the difference among groups, it is common to use the P n1 Pn1 +n2 Pn modified F test statistic T (x) = ( i=1 xi )2 /n1 +( i=n xi )2 /n2 +? ? ?+( i=n ng +1 xi )2 /ng , 1 +1 where n = n1 + n2 + ? ? ? + ngP . We can Pnrewrite the modified F statistic [3] as a second order n weighted v-statistic, i.e., T (x) = i1 =1 i2 =1 w(i1 , i2 )h(xi1 , xi2 ), here h(xi1 , xi2 ) = xi1 xi2 and w(i1 , i2 ) = 1/nk if both xi1 and xi2 belong to the k-th group, and w(i1 , i2 ) = 0 otherwise. The r-th moment of a resampling weighted v-statistic is: ? ? ? X E T r (x) = E w(i1 , ? ? ? , id )h(x ?i1 , ? ? ? , x =E

=

?

1 —R—

i1 ,??? ,id

X

i11 ,??? ,i1d ,??? ,ir1 ,??? ,ird

X? 2R

X

r n? Y

k=1

i11 ,??? ,i1d ,??? ,ir1 ,??? ,ird

w(ik1 , ? ? ? , ikd )

r n? Y

k=1

r ?? Y

h(x

k=1

w(ik1 , ? ? ? , ikd )

2

?id )

r ?? Y
k=1
?r
?o
,??? ?ik 1
,x
h(x
,??? ,x
?ik 1
) ?ik d
?ik d
)
?o
,
(1)

where is a resampling which is uniformly distributed in the whole resampling space R. —R—, the number of all possible resamplings, is equal to n! or nn for resampling without or with replacement. Thus the r-th moment of a resampling weighted v-statistic can be considered as a summation of products of data function terms and index function terms over a high-dimensional index set Udr = {1, ? ? ? , n}dr and all possible resamplings in R. Since both index space and resampling space are huge, it is computationally expensive for calculating resampling statistics directly. For terminology convenience, {(i11 , ? ? ? , i1d ), ? ? ? , (ir1 , ? ? ? , ird )} is called an index paragraph, which includes r index sentences (ik1 , ? ? ? , ikd ), k = 1, ? ? ? , r, and each index sentence has d index words ikj , j = 1, ? ? ? , d. Note that there are three different types of symmetry in computing resampling weighted v-statistics. The first symmetry is that permutation of the order of index words will not affect the result since the data function is assumed to be symmetric. The second symmetry is the permutation of the order of index sentences since multiplication is commutative. The third symmetry is that each possible resampling is equally likely to be chosen. In order to reduce the computational cost, first, the summation order is exchanged, r r ? ? n? Y ? ?Y X E T r (x) = w(ik1 , ? ? ? , ikd ) E h(x ?ik1 , ? ? ? , x where E

?Q
i11 ,??? ,i1d ,??? ,ir1 ,??? ,ird
r k=1
h(x
?ik 1
,??? ,x
?ik d
k=1
? ) =
1 —R—
k=1
P

2R
?Q
r k=1
h(x
?ik 1
,??? ,x
?ik d
? ) .
) ?ik d
?o
,
(2)

n {1, ? ? ? , n}dr = {(i11 , ? ? ? , i1d ), ? ? ? , (ir1 , ? ? ? , ird )}—ikm 2 o {1, ? ? ? , n}; m = 1, ? ? ? , d; k = 1, ? ? ? , r is then divided into disjoint index subsets, in ?Q ? r which E h(x is invariant. The above index set partition simplifies k , ? ? ? , x ?ik ) ?i k=1 1 d the ?computing of resampling ?statistics in the following sense: (a) we only need to calculate Qr E , ? ? ? , x ?ikd ) once per each index subset, (b) due to the symmetry of resamk=1 h(x ?ik 1 ?Q ? r pling, the calculation of E h(x is equivalent to calculating the average of k , ? ? ? , x ?ik ) ?i k=1 1 d all data function terms within the corresponding index subset, then we can completely replace all resamplings with simple summations, and (c) for further computational cost reduction, we can sort all index subsets in their symmetry order and calculate all index subset summations recursively. We will discuss the details in the following sections for both resampling without or with replacement. The abstract algebra terms used in this paper are listed as follows. The whole index set Udr
=

Terminology. A group is a non-empty set G with a binary operation satisfying the following axioms: closure, associativity, identity, and invertibility. The symmetric group on a set, denoted as Sn , is the group consisting of all bijections or permutations of the set. A semigroup has an associative binary operation defined and is closed with respect to this operation, but not all its elements need to be invertible. A monoid is a semigroup with an identity element. A set of generators is a subset of group elements such that all the elements in the group can be generated by repeated composition of the generators. Let X be a set and G be a group. A group action is a mapping G ? X ! X which satisfies the following two axioms: (a) e ? x 7! x for all x 2 X, and (b) for all a, b 2 G and x 2 X, a ? (b ? x) = (ab) ? x. Here the 0 ?0 denotes the action. It is well known that a group action defines an equivalence relationship on the set X, and thus provides a disjoint set partition on it. Each part of the set partition is called an orbit that denotes the trajectory moved by all elements within the group. We use symbol [ ] to represent an orbit. Two elements, x and y 2 X fall into the same orbit if there exists a g 2 G such that x = g ? y. The set of orbits is denoted by G X. A transversal of orbits is a set of representatives containing exactly one element from each orbit. In this paper, we limit our discussion to only finite groups [10,17].

3

Permutation

For permutation statistics, observations are permuted in all possible ways, i.e., R = Sn . Based on the three types of symmetry, we link the permutation statistics calculation with a group action. Definition 1. The action of G := Sn ? Sr ? Sd r on the index set Udr is defined as 3

( , ?, ?1 , ? ? ? , ?r ) ? ikm :=

? i??

k

1 1

?k , ?m

where m 2 {1, ? ? ? , d}, and k 2 {1, ? ? ? , r}.

Here, ?k denotes the permutation of the order of index words within the k-th index sentence, ? denotes the permutation of the order of r index sentences, and denotes the permutation of the value of an index word from 1 to n. For example, let n = 4, d = 2, r = 2, ?1 = ?1 1 = 1 ! 2, 2 ! 1, ?2 = ?2 1 = 1 ! 1, 2 ! 2, ? = ? 1 = 1 ! 2, 2 ! 1, and = 1 ! 2, 2 ! 4, 3 ! 3, 4 ! 1, then ( , ?, ?1 , ?2 ) ? {(1, 4)(3, 4)} = {(3, 1)(1, 2)} by {(1, 4)(3, 4)} ! {(4, 1)(3, 4)} ! {(3, 4)(4, 1)} ! {(3, 1)(1, 2)}. Note that the reason to define the action in this way is to guarantee G ? Udr ! Udr is a group action.

In most applications, both r and d are much less than the sample size n, we assume throughout this paper that n dr. ?Q ? r Proposition 1. The data function sum E , ? ? ? , x is invariant within each k) ?i k=1 h(x ?ik 1 d index orbit? of group action G := Sn ?? Sr ? Sd r acting on the index set Udr as defined in definition Qr , ? ? ? , x ?ikd ) = 1, and E k=1 h(x ?ik 1 Qr X k=1 h(xj1k , ? ? ? , xjdk ) ? ? , (3) 1 1 r r {(j11 ,??? ,jd1 ),??? ,(j1r ,??? ,jdr )}2[{(i11 ,??? ,i1d ),??? ,(ir1 ,??? ,ird )}] card [{(i1 , ? ? ? , id ), ? ? ? , (i1 , ? ? ? , id )}] ? ? where card [{(i11 , ? ? ? , i1d ), ? ? ? , (ir1 , ? ? ? , ird )}] is the cardinality of the index orbit, i.e., the number of indices within the index orbit [{(i11 , ? ? ? , i1d ), ? ? ? , (ir1 , ? ? ? , ird )}]. ?Q ? r Due to the invariance property of E h(x k , ? ? ? , x ?ik ) , the calculation of permutation ?i k=1 1 d statistics can be simplified by summing up all index function product terms in each index orbit. Proposition 2. The r-th moment of permutation statistics can be obtained by summing up the product of the data function orbit sum h and the index function orbit sum w over all index orbits, ? ? X w h E T r (x) = , (4) card([ ]) 2L

where = is a representative index paragraph, [ ] is the index orbit including , and L is a transversal of all index orbits . The data function orbit sum is {(i11 , ? ? ?

, i1d ), ? ? ?

, (ir1 , ? ? ?

, ird )}

X

h =

{(j11 ,??? ,jd1 ),??? ,(j1r ,??? ,jdr )}2[

r Y

] k=1
h(xj1k , ? ? ? , xjdk ),
(5)
and the index function orbit sum is w =
X
{(j11 ,??? ,jd1 ),??? ,(j1r ,??? ,jdr )}2[
r Y
] k=1
w(j1k , ? ? ? , jdk ).
(6)

Proposition 2 shows that the calculation of resampling weighted v-statistics can be solved by computing data function orbit sums, index function orbit sums, and cardinalities of all orbits defined in definition 1. We don?t need to conduct any real permutation at all. Now we demonstrate how to calculate orbit cardinalities, h and w . The following shows a naive algorithm to enumerate all index paragraphs and cardinality of each orbit of G Udr , which are needed to calculate h and w . We construct a Cayley Action Graph with a vertex set of all possible index paragraphs in Udr . We connect a directed edge from {(i11 , ? ? ? , i1d ), ? ? ? , (ir1 , ? ? ? , ird )} to {(j11 , ? ? ? , jd1 ), ? ? ? , (j1r , ? ? ? , jdr )} if {(j11 , ? ? ? , jd1 ), ? ? ? , (j1r , ? ? ? , jdr )} = gk {(i11 , ? ? ? , i1d ), ? ? ? , (ir1 , ? ? ? , ird )}, where gk is a generator 2 {g1 , ? ? ? , gp }. {g1 , ? ? ? , gp } is the set of generators of group G, i.e., G = hg1 , ? ? ? , gp i. It is sufficient and efficient to use the set of generators of group to construct the Cayley Action Graph, instead of using the set of all group elements. For example, we can choose {g1 , ? ? ? , gp } = { 1 , 2 } ? {?1 , ?2 } ? {?1 , ?2 }r , where 1 = (12 ? ? ? n), 2 = (12), ?1 = (12 ? ? ? r), ?2 = (12), ?1 = (12 ? ? ? d), and ?2 = (12). Here 1 = (12 ? ? ? n) denotes the permutation 1 ! 2, 2 ! 3, ? ? ? , n ! 1, and 2 = (12) denotes 4

1 ! 2, 2 ! 1, 3 ! 3, ? ? ? , n ! n. Note that listing the index paragraphs of each orbit is equivalent to finding all connected components in the Cayley Action Graph, which can be performed by using existing depth-first or breadth-first search methods [15]. Figure 1 demonstrates the Cayley Action Graph of G U21 , where d = 2, r = 1, and n = 3. Since the main effort here is to construct the Cayley Action Graph, the computational cost of the naive algorithm is O(ndr p) = O(ndr 22+r ). Moreover, the memory cost is O(ndr ). Unfortunately, this algorithm is not an offline one since we usually do not know the data size n before we have the data at hand, even d and r can be preset. In other words, we can not list all index orbits before we know the data size n. Moreover, since ndr 22+r is still computationally expensive, the naive algorithm is ill advised even if n is preset. i21 1
2
G U dr
3
1
G * U dr
*

i11 2 3 Cayley action graph
Set of orbits U 21 [{(1,1)}] [{(1,2)}]
Figure 1: Cayley action graph for G
?
U21 .
transversal
Figure 2: Finding the transversal.

In table 1, we propose an improved offline algorithm in which we assume that d and r are preset. For computing h and w , we find that we do not need to know all the index paragraphs within each index orbit. Since each orbit is well structured, it is enough to only list a transversal of orbits G Udr and corresponding cardinalities. For example, there are two orbits, [{(1, 1)}] and [{(1, 2)}], when d = 2 and r = 1. [{(1, 1)}], with cardinality n, includes all index paragraphs with i11 = i12 . 1 1 [{(1, 2)}], with cardinality n(n 1), n o includes all index paragraphs with i1 6= i2 . Actually, the transversal L = {(1, 1)}, {(1, 2)} carries all the above information. This finding reduces the computation cost dramatically. n Definition 2. We define an index set Udr ? = {1, ? ? ? , dr}dr = {(i11 , ? ? ? , i1d ), ? ? ? , (ir1 , ? ? ? , ird )}—ikm o 2 {1, ? ? ? , dr}; m = 1, ? ? ? , d; k = 1, ? ? ? , r and a group G? := Sdr ? Sr ? Sd r . Since we assumed n dr, Udr ? is a subset of the index set Udr . The group G? can be considered a subgroup of G since the group Sdr can be naturally embedded into the group Sn . Both Udr ? and G? are unrelated to the sample size n. Proposition 3. The transversal of G?
Udr ? is also a transversal of G
Udr .

By proposition 3, we notice that the listing of the transversal of G Udr is equivalent to the listing of the transversal of G? Udr ? (see Figure 2). The latter is computationally much easier than the former since the cardinalities of G? and Udr ? are much smaller than those of G and Udr when n dr. Furthermore, finding the transversal of G? Udr ? can be done without knowning sample size n. Due to the structure of each orbit of G Udr , we can calculate the cardinality of each orbit of G Udr with the transversal of G? Udr ? , although G Udr and G? Udr ? have different caridnalities for corresponding orbits. Table 1: Offline double sided searching algorithm for listing the transversal Input: d and r, 1. Starting from an orbit representative {(1, ? ? ? , d), ? ? ? , ((r 1)d + 1, ? ? ? , rd)} 2. Construct the transversal of Sdr Udr ? by merging 3. Construct the transversal of of G? Udr ? by graph isomorphism testing 4. Ending to an orbit representative {(1, ? ? ? , 1), ? ? ? , (1, ? ? ? , 1)} Output: a transversal L of G Udr , #( ), #( ! ?), and merging order(symmetry order) of orbits Comparing with the Cayley Action Graph naive algorithm, our improved algorithm lists the transversal of G Udr and calculates the cardinalities of all orbits more efficiently. In addition, the improved algorithm also assigns a symmetry order to all orbits, which helps further reduce the computational 5

cost of the data function orbit sum h and the index function orbit sum w . The base of our improved algorithm is on the fact that a subgroup acting on the same set causes a finer partition. On one hand, it is challenging to directly list

the transversal of G? Udr ? . On the other hand, it is much easier to find two related group actions, causing finer and coarser partitions of Udr ? . These two group actions help us find the transversal of G? Udr ? efficiently with a double sided searching method. Definition 3. The action of Sdr on the index set Udr ? is defined as Sdr , m 2 {1, ? ? ? , d}, and k 2 {1, ? ? ? , r}. Each orbit of Sdr [{(i11 , ? ? ? , i1d ), ? ? ? , (ir1 , ? ? ? , ird )}]s .

? ikm , where 2 Udr ? is denoted by

Note the group action defined in definition 3 only allows permutation of index values, it does not allow shuffling of index words within each index sentence or of index sentences. Since Sdr is embedded in G? , the set of orbits Sdr Udr ? is a finer partition of G? Udr ? . For example, both [{(1, 2)(1, 2)}]s and [{(1, 2)(2, 1)}]s are finer partitions of [{(1, 2)(1, 2)}]. In addition, it is easy to construct a transversal of Sdr Udr ? by merging distinct index elements. Definition 4. Given a representative I, which includes at least two distinct index values, for example i 6= j, an operation called merging replaces all index values of i or j with min(i, j). For example, [{(1, 2)(2, 3)}] becomes [{(1, 1)(1, 3)}] after merging the index values of 1 and 2. ?

1

?(k,m)

Definition 5. The action of Sdr ?Sdr on the index set Udr ? is defined as ?i? 1 ?(k,m)sw , where ? 2 Sdr denotes a permutation of all dr index words without any restriction, i.e. ? 1 ? (k, m)s denotes the index sentence location after permutation ?, and ? 1 ? (k, m)w denotes the index word location after permutation ?. The orbit of Sdr ? Sdr Udr ? is denoted by [{(i11 , ? ? ? , i1d ), ? ? ? , (ir1 , ? ? ? , ird )}]l .

Since the group action defined in definition 5 allows free shuffling of the order of all dr index words, the order does not matter for Sdr ? Sdr Udr ? and shuffling can across different sentences. For example, [{(1, 2)(1, 2)}]l = [{(1, 1)(2, 2)}]l . Sdr ?Sdr Udr ? is a coarser partition of G? Udr ? . Proposition 4. A transversal of Sdr Udr ? can be generated by all possible mergings of s [{(1, ? ? ? , d), ? ? ? , (d(r 1) + 1, ? ? ? , dr)}] . Proposition 5. Enumerating a transversal of Sdr ? Sdr of dr.

Udr ? is equivalent to the integer partition

We start the transversal graph construction from an initial orbit [{(1, ? ? ? , d), ? ? ? , (d(r 1) + 1, ? ? ? , dr)}]s , i.e, all index elements have distinct values. Then we generate new orbits of Sdr Udr ? by merging distinct index values in existing orbits until we meet [{(1, ? ? ? , 1), ? ? ? , (1, ? ? ? , 1)}]s , i.e., all index elements have equal values. We also add an edge from an existing orbit to a new orbit generated by merging the existing one. The procedure for d = 2, r = 2 case is shown in Figure 3. Now we generate the transversal of G? Udr ? from that of Sdr Udr ? . This can be done by checking whether two orbits in Sdr Udr ? are equivalent in G? Udr ? . Actually, orbit equivalence checking is equivalent to the classical graph isomorphism problem since we can consider each index word as a vertex and connect two index words if they belong to the same index sentence. The graph ? isomorphism ? testing can be done by Luks?s famous algorithm [1,15] with computational p ? cost exp O( vlogv)

, where v is the number of vertices. Figure 4 shows a transversal of G? U22 ?

generated from that of S4 U22 (Figure 3). By proposition 3, it is also a transversal of G U22 . Since G? Udr ? is a finer partition of Sdr ? Sdr Udr ? , orbit equivalence testing is only necessary when two orbits of Sdr Udr ? correspond to the same integer partition. This is why we named this algorithm double sided searching. [{(1,2)(3,4)}]

[{(1,2)(3,4)}]s

[{(1,1)(2,3)}]

[{(1,1)(3,4)}]s [{(1,2)(1,4)}]s [{(1,2)(3,1)}]s [{(1,2)(2,4)}]s [{(1,2)(3,2)}]s [{(1,2)(3,3)}]s [{(1,1)(1,4)}]s [{(1,1)(3,1)}]s [{(1,1)(3,3)}]s [{(1,2)(1,1)}]s [{(1,2)(1,2)}]s [{(1,2)(2,1)}]s [{(1,2)(2,2)}]s

[{(1,1)(1,2)}]

[{(1,2)(1,3)}]

[{(1,1)(2,2)}] [{(1,1)(1,1)}]

[{(1,1)(1,1)}]s

Figure 3: Transversal graph for S4 [(1,2) (3,4)] [(1,1)(3,4)]

[(1,2)(1,4)]

[(1,2)(3,1)]

[(1,2)(2,4)]

Figure 4: graph for G

?

U22 . [(1,2)(3,2)]

[(1,2)(3,3)]

6

[(1,1)(1,4)] [(1,1)(3,1)] [(1,1)(3,3)] [(1,2)(1,1)] [(1,2)(1,2)] [(1,2)(2,1)] [(1,2)(2,2)]s

Transversal U22 .

[{(1,2)(1,2)}]

Definition 6. For any two index orbit representatives 2 L and ? 2 L, we say that ? has a lower merging or symmetry order than that of , i.e., ? , if [?] can be obtained from [ ] by several mergings. Or there is a path from [ ] to [?] in the transversal graph. Here L denotes a transversal set of all orbits. Definition 7. We define #( ) as the number of Sdr Udr ? orbits in [ ]. We also define #( ! ?) as the number of different [?]s s which can be reached from a [ ]s . It is easy to get #( ) when we generate a transversal graph of G Udr from that of Sdr Udr ? . The #( ! ?) can also be obtained from the transversal graph of G Udr by counting the number of different [?]s s which can be reached from a [ ]s . For example, there are edges connecting [{(1, 1)(3, 4)}]s to [{(1, 1)(1, 4)}]s and [{(1, 1)(3, 1)}]s . Since [{(1, 1)(1, 4)}] = [{(1, 1)(3, 1)}] = [{(1, 1)(1, 2)}], #( = {(1, 1)(2, 3)} ! ? = {(1, 1)(1, 2)}) = 2. Note that this number can also be obtained from [{(1, 2)(3, 3)}]s to [{(1, 2)(1, 1)}]s and [{(1, 2)(2, 2)}]s . The difficulty for computing data function orbit sum and index function orbit sum comes from two constraints: equal constraint and unequal constraint. For example, in the orbit [{(1, 1), (2, 2)}], the equal constraint is that the first and the second index values are equal and the third and fourth index values are also equal. On the other hand, the unequal constraint requires that the

first two index values are different from the last two. Due to the difficulties mentioned, we solve this problem by first relaxing the unequal constraint and then applying the principle of inclusion and exclusion. Thus, the calculation of an orbit sum can be separated into two parts: the relaxed orbit sum without unequal constraint and lower order orbit sums. For example, the relaxed index function orbit sum is ?P ?2 P w? =[{(1,1),(2,2)}] = i,j w(i, i)w(j, j) = w(i, i) . i Proposition 6. The index function orbit sum w can be calculated by subtracting all lower order orbit sums from the corresponding relaxed index function orbit sum w? , i.e., w = w? P ) w? #( 1) ? ? ? (n q + 1), where q is the ? #(?) #( ! ?). The cardinality of [ ] is #( )n(n number of distinct values in . The calculation of the data index function orbit sum h is similar.

So the computational cost mainly depends on the calculation of relaxed orbit sum and the lowest order orbit sum. The computational cost of the lowest order term is O(n). The calculation of relaxed orbit can be done by Zhou?s greedy graph search algorithm [21]. Proposition 7. For d 2, let m(m 1)/2 ? rd(d 1)/2 ¡ (m + 1)m/2, where r is the order of moment and m is an integer. For a d-th order weighted v-statistic, the computational cost of the orbit sum for the r-th moment is bounded by O(nm ). When d = 1, the computational complexity of the orbit sum is O(n).

4

Bootstrap

Since Bootstrap is resamping with replacement, we need to change Sn to the set of all possible endofunctions Endn in our computing scheme. In mathematics, an endofunction is a mapping of a set to its subset. With this change, H := Endn ? Sr ? Sdr acting on Udr becomes a monoid action instead of a group action since endofunction is not invertible. The monoid action also divides the Udr into several subsets. However, these subsets are not necessarily disjoint after mapping. For example, when d = 2 and r = 1, we can still divide the S index set U21 into two subsets, i.e., [(1, 1)] and [(1, 2)]. However, [(1, 2)] is mapped to U21 = [(1, 2)] [(1, 1)] by monoid action H ? Udr ! Udr , although [(1, 1)] is still mapped to itself. Fortunately, the computation of Bootstrap weighted v-statistics only needs index function orbit sums and relaxed data function orbit sums in the corresponding permutation computation. Therefore, the Bootstrap weighted v-statistics calculation is just a subproblem of permutation weighted v-statistics calculation. Proposition 8. We can obtain the r-th moment of bootstrapping weighted v-statistics by summing up the product of the index function orbit sum w and the relaxed data function orbit sum h? over all index orbits, i.e., X w h? E (T r (x)) = , (7) card([ ? ]) where

2 Endn , card([

?

2L

]) = #( )n , and q is the number of distinct values in . q

7

Table 2: Comparison of accuracy and complexity for calculation of resampling statistics.

Linear Permutation Quadratic Linear Bootstrap Quadratic

Methods Exact Our Random Exact Our Random Exact Our Random Exact Our Random

2nd moment 0.7172 0.7172 0.7014 1.0611e3 1.0611e3 1.0569e3 3.5166 3.5166 3.4769 2.4739e5 2.4739e5 2.4576e5

3rd -0.8273 -0.8273 -0.8326 -4.6020e4 -4.6020e4 -4.5783e4 8.9737 8.9737 8.8390 -6.0322e6 -6.0322e6 -5.9825e6

4th 1.0495 1.0495 1.0555 2.1560e6 2.1560e6 2.1825e6 35.4241 35.4241 34.6393 2.6998e8 2.6998e8 2.6589e8

Time 1.1153e3 0.0057 0.5605 1.718e3 0.006 2.405 204.4381 0.0053 0.3294 445.536 0.005 1.987

The computational cost of bootstrapping weighted v-statistics is the same level as that of permutation statistics.

5

Numerical results

To evaluate the accuracy and efficiency of our mothds, we simulated data and conduct perPgenerate n mutation and bootstrapping for both linear test statistic w(i)h(x i ) and quadratic test statistic i=1 Pn Pn w(i , i )h(x , x ) . To demonstrate the universal applicability of our method and 1 2 i i 1 2 i1 =1 i2 =1 prevent a chance result, we generate w(i), h(xi ), w(i1 , i2 ), h(xi1 , xi2 ) randomly. We compare the accuracy and complexity among exact permutation/bootstrap, random permutaton/bootrap (10,000 times), and our methods. Table 2 shows comparisons for computing the second, third, and fourth moments of permutation statistics with 11 observations (the running time is in seconds) and of bootstrap statistics with 8 observations. In all cases, our method achieves the same moments as those of exact permutation/bootstrap, and reduces computational cost dramatically comparing with both random sampling and exact sampling. For demonstration purpose, we choose a small sample size here, i.e., sample size is 11 for permutation and 8 for bootstrap. Our method is expected to gain more computational efficiency as n increases.

6

Conclusion

In this paper, we propose a novel and computationally fast algorithm for computing weighted vstatistics in resampling both univariate and multivariate data. Our theoretical framework reveals that the three types of symmetry in resampling weighted v-statistics can be represented by a product of symmetric groups. As an exciting result, we demonstrate the calculation of resampling weighted v-statistics can be converted into the problem of orbit enumeration. A novel efficient orbit enumeration algorithm has been developed by using a small group acting on a small index set. For further computational cost reduction, we sort all orbits by their symmetry order and calculate all index function orbit sums and data function orbit sums recursively. With computational complexity analysis, we have reduced the computational cost from n! or nn level to low-order polynomial level.

7

Acknowledgement

This research was supported by the Intramural Research Program of the NIH, Clinical Research Center and through an Inter-Agency Agreement with the Social Security Administration, the NSF CNS 1135660, Office of Naval Research award N00014-12-1-0125, Air Force Office of Scientific Research award FA9550-12-1-0201, and IC Postdoctoral Research Fellowship award 201111071400006. 8

# 2 References

[01] Babai, L., Kantor, W.M. , and Luks, E.M. (1983), Computational complexity and the classification of finite simple groups, Proc. 24th FOCS, pp. 162-171. [02] Minaei-Bidgoli, B., Topchy, A., and Punch, W. (2004), A comparison of resampling methods for clustering ensembles, In Proc. International Conference on Artificial Intelligence, Vol. 2, pp. 939-945. [03] Estabrooks, A., Jo, T., and Japkowicz, N. (2004), A Multiple Resampling Method for Learning from Imbalanced Data Sets, Comp. Intel. 20 (1) pp. 18-36. [04] Francois, D., Rossib, F., Wertza, V., and Verleysen, M. (2007), Resampling methods for parameter-free and robust feature selection with mutual information, Neurocomputing 70(7-9):1276-1288. [05] Good, P. (2005), Permutation, Parametric and Bootstrap Tests of Hypotheses, Springer, New York. [06] Gretton, A., Borgwardt, K., Rasch, M., Scholkopf, B., and Smola, A. (2007), A kernel method for the two-sample- problem, In Advances in Neural Information Processing Systems (NIPS). [07] Guo, S. (2011), Bayesian Recommender Systems: Models and Algorithms, Ph.D. thesis. [08] Hopcroft, J., and Tarjan, R. (1973), Efficient algorithms for graph manipulation, Communications of the ACM 16: 372-378. [09] Huang, J., Guestrin, C., and Guibas, L. (2007), Efficient Inference for Distributions on Permutations, In Advances in Neural Information Processing Systems (NIPS). [10] Kerber, A. (1999), Applied Finite Group Actions, Springer-Verlag, Berlin. [11] Kondor, R., Howard, A., and Jebara, T. (2007), Multi-Object Tracking with Representations of the Symmetric Group, Artificial Intelligence and Statistics (AISTATS). [12] Kuwadekar, A. and Neville, J. (2011), Relational Active Learning for Joint Collective Classification Models, In International Conference on Machine Learning (ICML), P. 385-392. [13] Liu, H., Palatucci, M., and Zhang, J.(2009), Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery, In International Conference on Machine Learning (ICML). [14] Matthew Higgs and John Shawe-Taylor. (2010), A PAC-Bayes bound for tailored density estimation, In Proceedings of the International Conference on Algorithmic Learning Theory (ALT). [15] McKay, B. D. (1981), Practical graph isomorphism, Congressus Numerantium 30: 45-87, 10th. Manitoba Conf. on Numerical Math. and Computing. [16] Mielke, P. W., and K. J. Berry (2007), Permutation Methods: A Distance Function Approach, Springer, New York. [17] Nicholson, W. K. (2006), Introduction to Abstract Algebra, 3rd ed., Wiley, New York. [18] Serfling, R. J. (1980), Approximation Theorems of Mathematical Statistics, Wiley, New York. [19] Song, L. (2008), Learning via Hilbert Space Embedding of Distributions, Ph.D. thesis. [20] Sutton, R. and Barto, A. (1998), Reinforce-

ment Learning, MIT Press. [21] Zhou, C., Wang, H., and Wang, Y. M. (2009), Efficient moments-based permutation tests, In Advances in Neural Information Processing Systems (NIPS), p. 2277-2285.

9