Iterated Tverberg Point Algorithm

Yakun Li
2690390

Lab Development and Application of Data Mining and Learning Systems
Summer Semester 2015

Rheinische Friedrich-Wilhelms-Universität Bonn
Fraunhofer IAIS, Sankt Augustin

Abstract. In some machine learning tasks, we are only able to learn sets of models from portions of the dataset, like distributed learning or learning on mobile devices. A classical approach to get a stronger model from a set of models is averaging them. In this lab, we use Iterated Tverberg point algorithm to combine a set of models to a new model. We empirically evaluate this approach in comparison to a hypothetical centrally computed model as well as to the average of weak models. From the experiments, it turns out that the Iterated Tverberg point algorithm can combine a set of weak models to a much better model.

1 Introduction

In this lab, we tackle the problem of combing a set of linear models obtained by machine learning algorithms to create a better model. Because in some machine learning tasks, the dataset used for training is huge or we are only able to get subsets of it. It will take too much time to learn a good model from the whole dataset or it is not possible. Assume that we are able to learn a mount of weak models from comparatively smaller subsets, and combine them to a new model. If the combined model has a good performance and takes much less time to be computed, then it will be a good approach, and approaches of this kind can be found in boosting, as well as in distributed computation.

A center point is a natural generalization of median to higher dimensions, and the Iterated Tverberg point algorithm computes an approximate center point. The Iterated Tverberg point algorithm is the first deterministic algorithm which computes an approximate center point of a set S of n points in R^d with run time sub-exponential in d. Here, we apply the algorithm to combine a set of models learned by machine learning algorithm to a much better model. In the context of this lab, a point refers to a model learned from a dataset.

In this lab, we use the Iterated Tverberg point algorithm to combine a set of linear models, each of which has been created by a Machine Learning algorithm. Then we compare the performances of the model learned on the whole

dataset, the models learned on the subsets, the averaged model and the combined model. Section 2 will introduce some background knowledge which is helpful to understand the algorithm. Section 3 will introduce the Iterated Tverberg point algorithm. Some experiments testing the performance of the algorithm will be given in Section 4. The last Section 5 will give a summary of the lab.

2 Related Work and Preliminaries

The Iterated Tverberg point algorithm was first introduced by Miller and Sheehy [2008]. A good explanation of the basic concepts and algorithm can be found in Sheehy [March, 2010], Nguyen [Sep, 2008]. This section will introduce some background knowledge to understand the algorithm. The background knowledge and algorithm parts will mainly reference Miller and Sheehy [2008], Sheehy [March, 2010], and proofs of the theorems which are closely related to the implementation will mainly reference Nguyen [Sep, 2008].

The following in this section contains some definitions and theorems which are helpful to understand the Iterated Tverberg point algorithm.

Definition 1 *Closed Halfspace is a set* $\{x \in R^d : a \cdot x \leq b, a \in R^d, b \in R\}$*Sheehy [March, 2010].*

Definition 2 *The Tukey depth of a point x with respect to a set P is the minimum number of points of P in any halfspace containing xSheehy [March, 2010], like figure 1.*

Fig. 1: depth of blue point w.r.t black points set is 1.

This report will call *Tukey depth* as *depth* for brevity. We are trying to pick a point of maximum depth to approximate median in high dimensions.

Definition 3 *A point x is a centerpoint with respect to a set* $P \subset R^d$ *if depth of p is at least* $\frac{n}{d+1}$*Sheehy [March, 2010].*

There is a point of depth at least $\frac{n}{d+1}$ for any $P \subset R^d$ Miller and Sheehy [2008], so centerpoints always exist.

Definition 4 *Convex hull of a set A, denoted by conv(A), is the set of all convex combination of points in AMiller and Sheehy [2008]. That is:*

$$conv(A) = \{x | \exists a_1, ..., a_N \in A, \alpha_1, ..., \alpha_n \geq 0; \sum_{i=1}^{N} \alpha_i = 1; x = \sum_{i=1}^{N} \alpha_i a_i.\} \quad (1)$$

Theorem 1 *(Radon). Let $a_1, a_2, ..., a_m \in R^d$, $m \geq d+2$, then there is a partition of 1,...,m into I and J such that the convex hull of $\{a_i, i \in I\}$ and $\{a_j, j \in J\}$ is nonemptyNguyen [Sep, 2008].*

Proof. Denote $b_i := (a_i, 1)$. There are $m \geq d+2$ vectors in $d+1$ dimensional space, thus they are linearly dependent. It means that there exists $\alpha_1, ..., \alpha_m$ not all zero such that:

$$\sum_{i=1}^{m} \alpha_i b_i = 0. \tag{2}$$

Construct $I = \{i|\alpha_i \geq 0\}$ and $J = \{j|\alpha_j < 0\}$. As the last coordinate of b_i is 1, thus:

$$\begin{aligned} \sum_{i \in I} \alpha_i + \sum_{j \in J} \alpha_j = 0. \\ \sum_{i \in I} \alpha_i = -\sum_{j \in J} \alpha_j = \alpha \neq 0. \end{aligned} \tag{3}$$

Then the vector

$$\sum_{i \in I} \frac{\alpha_i}{\alpha} a_i = -\sum_{j \in J} \frac{\alpha_j}{\alpha} a_j \tag{4}$$

is in both the convex hull of $\{a_i|i \in I\}$ and $\{a_j|j \in J\}$, the theorem 1 is proved.

Theorem 2 *(Carathéodory). For $S \subset R^d$, if $x \in conv(S)$ then $x \in conv(R)$ for some $R \subset S$, $|R| \leq d+1$Nguyen [Sep, 2008].*

Proof. Assume that $x = \sum_{i|x_i \in S} \alpha_i x_i$, with $\alpha_i > 0$; $\sum_i \alpha_i = 1$ and $|S| > d+1$. Then there exists a set S' of size smaller than $|S|$, with $x \in conv(S')$. Thus, we are able to reduce the size of the set whose convex hull contains x until we get R of size at most d+1.

Denote $x_2 - x_1, x_3 - x_1, ..., x_{d+2} - x_1$, there are $d+1$ vectors in d-dimension, thus they are linearly dependent. Therefore there exists $\beta_2, ..., \beta_{d+2}$ not all zero such that:

$$\begin{aligned} \sum_{i=2}^{d+2} \beta_i (x_i - x_1) = 0. \\ \sum_{i=2}^{d+2} \beta_i x_i = \sum_{i=2}^{d+2} \beta_i x_1. \end{aligned} \tag{5}$$

Let $\beta_1 = -\sum_{i \geq 2} \beta_i$, $\beta_j = 0$ for $j > d+2$, then

$$\begin{aligned} \sum_{i=1}^{d+2} \beta_i x_i = 0, \sum_{j=d+3} \beta_j x_j = 0. \\ \sum_i \beta_i x_i = 0 \; and \; \sum_i \beta_i = 0. \end{aligned} \tag{6}$$

Then,

$$x = \sum_i \alpha_i x_i = \sum_i \alpha_i x_i - \lambda \sum_i \beta_i x_i \ for\ all\ \lambda \tag{7}$$

$$x = \sum_i (\alpha_i - \lambda\beta_i) x_i \ for\ all\ \lambda \tag{8}$$

Thus, we can choose a λ such that all $\alpha' = \alpha_i - \lambda\beta_i \geq 0$ and at least one such value is 0. As $\alpha'_i \geq 0, \sum_i \alpha'_i = \sum_i \alpha_i = 1$, we get another convex representation of x with support size smaller than $|S|$. Thus, the theorem 2 is proved.

3 Iterated Tverberg Point Algorithm

The Iterated Tverberg point algorithm is algorithm 1 as shown below.

input : $S \in R^d : |S| = n$

Initialize empty stacks $B_0, ..., B_{lg\lceil \frac{n}{2(d+1)^2} \rceil}$;
Push $< s, \{s\} >$ to B_0 for each $s \in S$;
while $B_{lg\lceil \frac{n}{2(d+1)^2} \rceil}$ *is empty* **do**
    Initialize proof to be an empty stack ;
    let l be the max such that B_{l-1} has at least d + 2 points;
    Pop d + 2 points $q_1, ..., q_{d+2}$ from B_{l-1};
    Let $< c, \{U_1, U_2\} >$ = RADON($q_1, ..., q_{d+2}$);
    for $k = 1, 2$ **do**
        for $i = 1, ..., 2^{l-1}$ **do**
            Let S_{ij} be the ith part of the proof for q_j;
            Let $X = \cup_{q_j \in U_k} S_{ij}$;
            Let X' = PRUNE(c,X);
            Push X' to proof;
            Push $(< s, \{s\} >)$ to B_0 for each $x \in X \backslash X'$;
        end
    end
    Push $< c, proof >$ to B_l ;
end
Return $< c, proof >$ from $B_{lg\lceil \frac{n}{2(d+1)^2} \rceil}$;

Algorithm 1: Iterated Tverberg point AlgorithmMiller and Sheehy [2008]

Definition 5 *Given a point $p \in R^d$, a proof that p has Turkey depth (at least) r with respect to S is a collection of r disjoint subsets $U_1, U_2, ..., U_r \subset S$ such that $p \in \cap_{j=1}^{r} conv(U_j)$Miller and Sheehy [2008], like in figure 2.*

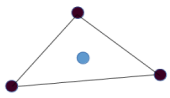


Fig. 2: proof of Tukey depth of blue point w.r.t black points set is 1.

If such a proof exists, then p has depth r. For example, any point $s \in S$, the set $\{s\}$ is a proof of depth 1. In the algorithm 1, a point and its proof of depth is denoted by $< point, proof >$, and a collection of proofs are disjoint if their sets are pairwise disjoint. The algorithm will return a point of depth at least $\left\lceil \frac{n}{2(d+1)^2} \right\rceil$ as shown in theorem 3, so we initialize B_i to be empty to store points of depth 2^i in the step 1 of the algorithm, for $\forall i \in \{0, ..., lg \left\lceil \frac{n}{2(d+1)^2} \right\rceil\}$. The steps 2 and 3 of the algorithm initialize B_0 with points of depth 1, and iterate the while loop until we get a point of desired depth. The steps 4 and 5 in 1 initialize the proof and get the index l, with at least $d+2$ points in B_{l-1}. Then we are able to pop out $d+2$ points from B_{l-1}, and apply theorem 1 to get Radon point c, partitions U_1 and U_2 in step 6 and 7. After that, we are able to iterate the two partition U_1 and U_2 from step 8, and start to iterate the parts of proof of each point in U_1 and U_2 from step 9. In steps 10 and 11, we form X as a proof that c has depth at least 2^l. In steps 12 and 13, we prune the size of X to get X' as shown in theorem 2, and push X' to the proof. Then we push the pruned points to B_0 in step 14. When the algorithm terminates, we return $< c, proof >$ from $B_{lg\left\lceil \frac{n}{2(d+1)^2} \right\rceil}$. The run time of Iterated Tverberg Point algorithm is $n^{O(lgd)}$, which consists of $n^{O(logd)}$ number of iterations of the main loop, and $O(nd^3)$ run time of each iterationMiller and Sheehy [2008].

Theorem 3 *The Iterated Tverberg point Algorithm returns a point of depth at least* $\left\lceil \frac{n}{2(d+1)^2} \right\rceil$ *Miller and Sheehy [2008].*

Proof. It can be seen from the algorithm that it returns a point of the desired depth and its corresponding proof if it terminates. Assume that the algorithm does not terminate. Which means that it is impossible to pick l at some time t, because every bucket has fewer than $d+2$ points. Let h be the maximum such that B_h is nonempty at time t. For any point p in a bucket B_i, with $i \in \{0, ..., h\}$, the proof for p contains at most $2^i(d+1)$ input points, with $r = 2^i$. Since all of the proofs in all of the buckets are disjoint, so

$$\sum_{i=0}^{h} 2^i (d+1)^2 \geq n. \tag{9}$$

Therefore,

$$2^{h+1} - 1 \geq \frac{n}{(d+1)^2}. \tag{10}$$

Thus,

$$2^h > \frac{n}{2(d+1)^2}. \tag{11}$$

Then it implies that $h > lg\frac{n}{2(d+1)^2}$, and $B_{lg\left\lceil \frac{n}{2(d+1)^2} \right\rceil}$ is nonempty. However, this would have caused the algorithm to break out of the while loop and terminate, contrary to our assumption. Thus, the theorem 3 is proved.

4 Experiments

This section will introduce some performance tests of the Iterated Tverberg point algorithm. The general settings of the experiments are like in figure 3, and with n instances in Dataset, m instances in each subset, with $m \leq n$.

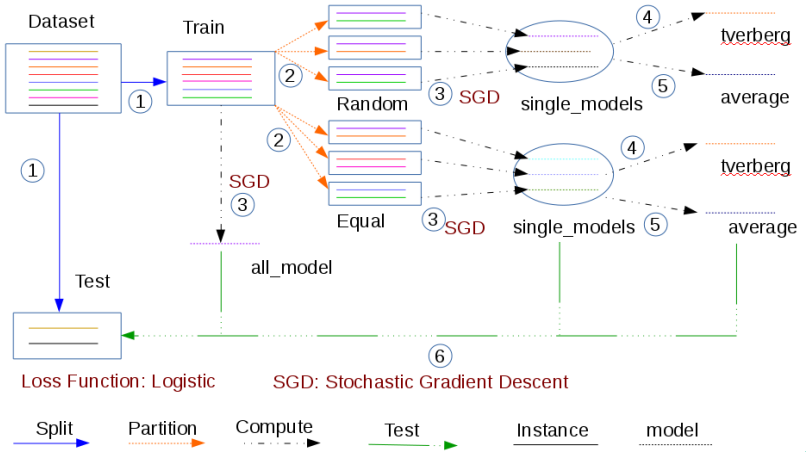


Fig. 3: experiments setting with 3 subsets.

1. Randomly split dataset into two subsets for training and testing.
2. Randomly partition instances from training dataset into subsets, alternatively select instances from training dataset to form equal size disjoint subsets.
3. Learn models from subsets and whole training dataset.
4. Combine set of single models into Tverberg point.
5. Combine set of single models by averaging them.
6. Test all models and points against test dataset.

We use Stochastic Gradient Descent(SGD) algorithm from *Sklearn* python library to learn single models, and logistic function as the loss function. The tests

of the algorithm are mainly performed from three aspects, which are included in section 4.1, 4.2, 4.3. The tests of varying features and varying number of models are performed on synthetic data generated by $datasets.make\_classification$ from *Sklearn* library. The test of varying instances is performed on the dataset from UCI machine learning repository. The datasetTel [May, 2007] is generated to simulate registration of high energy gamma particles in an atmospheric telescope.

4.1 Varying number of features test

The test of different number of features is performed on synthetic dataset, and the range of features is from 2 to 14, and 1 feature is added at each time. The dataset for each number of features is generated once, so the equally disjoint datasets are also generated once in each number of features test. The number of models are 1000, and the general settings are included in table 1, and some running results are also given in the following parts.

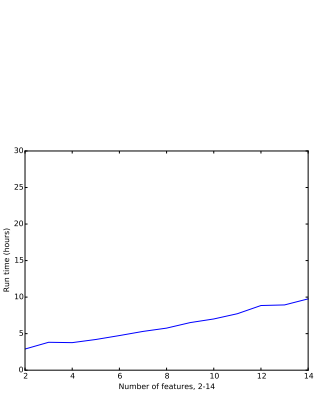
Synthetic Dataset	
Number of instances	500000, 99.6% training, 0.4% testing
Number of models each experiment	1000
Number of features	2-14, with step 1

Table 1: Synthetic dataset for different number of features test

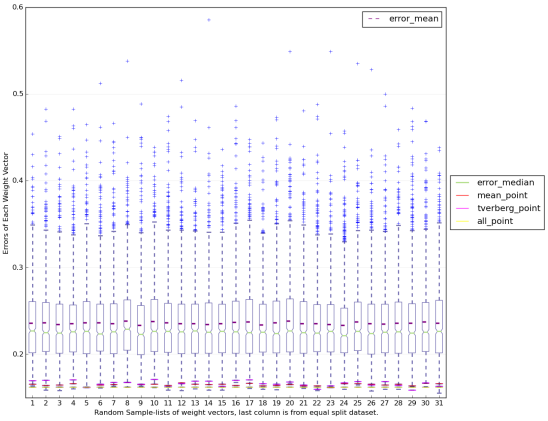
It can be seen from figure 4a that the run time of the experiments is increasing almost linearly with respect to the increasing number of features. We can also see from the algorithm 1 that we need to have $n > 2(d+1)^2$ number of models to compute a Tverberg point with depth lager than one.

The figure 4b represents the experiment result generated by the models learned from the subsets containing 10 features. The last column is the result of the models learned from equally disjoint subset. It can be seen that the error of all point is the lowest most of the time. There is no significant difference between the performance of tverberg point and mean point as shown in figures 4b and 4c.

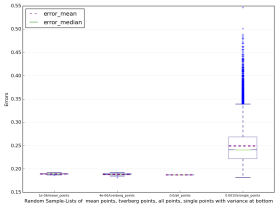
However, the test error of the tverberg point and mean point with respect to the increasing number of features is not smooth, as shown in figures 4d and 4e. The reason should be that the performances of the weak models in each dimension test are not guaranteed, so the performance of the combined Tverberg point is not guaranteed. And these two figures are the testing results from 2 to 29 features, because it might be too short to see the trend of the error from just 2 to 14 features.



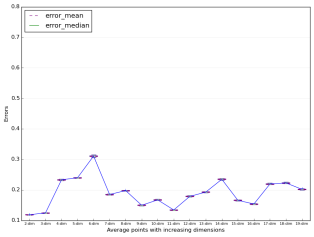
(a) Run time w.r.t increasing number of features.



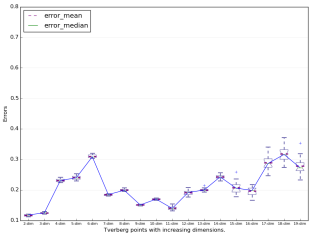
(b) Errors w.r.t 30 experiments.



(c) Errors w.r.t 30 experiments.



(d) Errors of Average point w.r.t 2-19 dimensions.



(e) Errors of Tverberg point w.r.t 2-19 dimensions.

Fig. 4: Varying number of features test.

4.2 Varying number of weak models test.

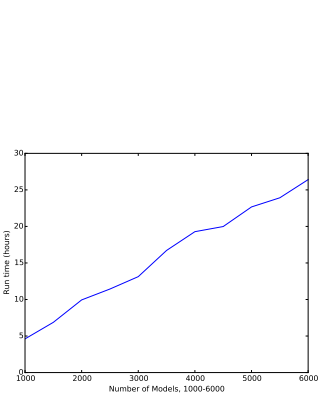
The test of different number of single models is performed on synthetic dataset, and the number of different models is from 1000 to 6000, and 500 models are added at each time. The number of features of the generated synthetic dataset is 5 and the general setting is included in table 2.

Synthetic Dataset	
Number of instances	500000, 99.6% training, 0.4% testing
Number of features of dataset	5
Number of weak models	1000-6000, with step 500

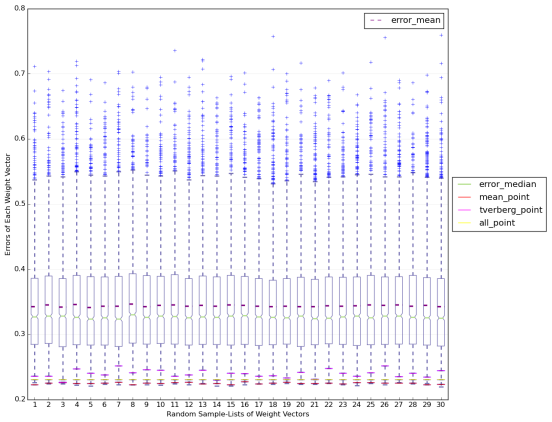
Table 2: Synthetic dataset for different number of weak models test

The running results are plotted in figure 5, we are able to see from 5a that the run time is increasing almost linearly with the increasing number of models. We can see from 5b and 5c that the performances of the Tverberg point and average point are better than most of the weak models, and the performance of the all

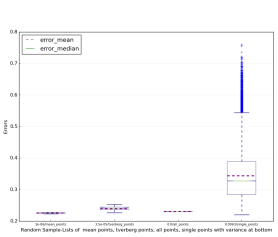
point is very good. The error of the Average point is almost keeping steadily with the increasing number of models as shown in 5d. The error of the Tverberg point is increasing with the increasing number of weak models as shown in 5e.



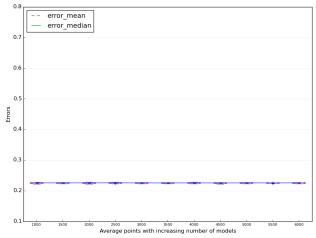
(a) Run time w.r.t increasing number of weak models.



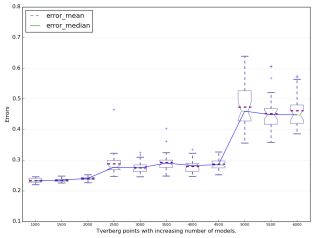
(b) Example taken from 2000 weak models.



(c) Example taken from 2000 weak models.



(d) Errors of Average point w.r.t 1000-6000 weak models.



(e) Errors of Tverberg point w.r.t 1000-6000 weak models.

Fig. 5: Varying number of weak models test.

4.3 Varying number of instances for single model test

The dataset descriptions are in table 3, and the settings of the experiments are in table4. We can see from table 3 that there are 19020 instances in total, so we are not able to select 1000 equally disjoint subsets from this dataset, otherwise, there are too few instances for SGD learning for each single model. The testing of the algorithm on this dataset is only performed on randomly selected instances with 1000 subsets.

One of the test example is in figure 6b, the test is performed on 1400 instances for each single model learning. We can see from the figure that the performance of the Tverberg point is very good comparing with other single models. The all point learned from the whole dataset sometimes has a very low error, however it has a very high error also occasionally. The performance of the mean point is

MAGIC Gamma Telescope Data Set		
Number of features	10	
Labels	g/h	gamma / hadron
Number of Instances	19020	12332(g) / 6688(h)

Table 3: Data Set

Experiments Settings	
Different Number of Instances	100 - 2900, step 100
Number of experiments each instance number	30
Number of models each experiment	1000

Table 4: Experiments Settings

also very good, and we can see from figure 6a that the performance of Tverberg and mean point is comparable.

We are able to see from figure 6c that the mean of error of average point is decreasing with increasing number of instances for each single model learning. We can also see this from figure 6d for Tverberg point. However, the learning time with increasing number of instances for each single model learning is not increasing dramatically, which can be seen from figure 6e.

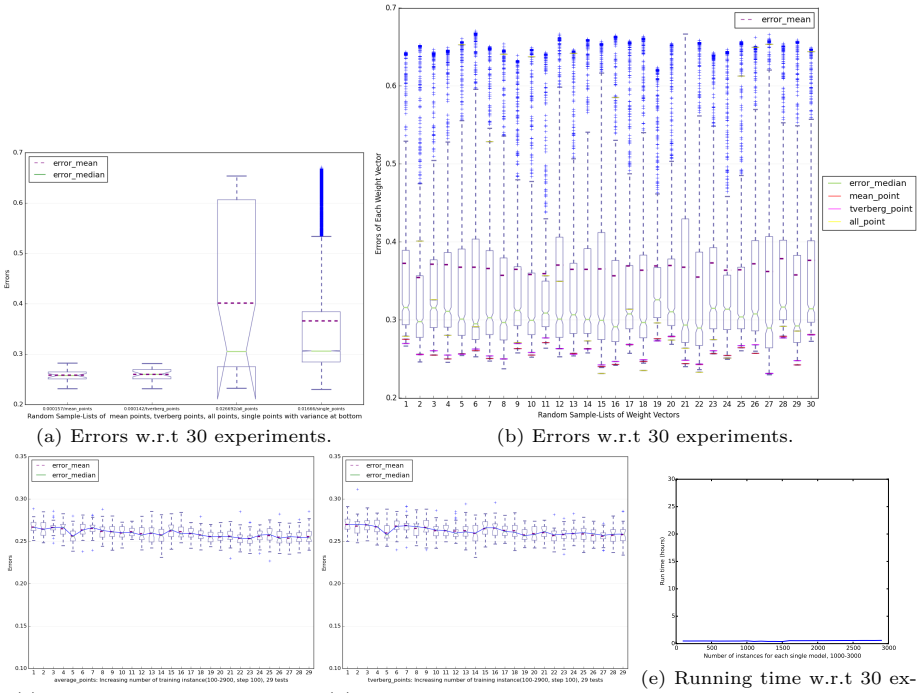
4.4 Test Comparison

The results in figures 4b, 5b and 6b show that the Iterated Tverberg Point algorithm can combine a set of models learned by machine learning algorithms to a much better model.

From Section 4.1, we see that when the dataset has a very high number of features, then a lot of single models are needed to compute the Tverberg point. With the increasing number of single models are computed, the running time is increasing quickly which can be seen from figure 5a. In order to compute more single models, we also need a sufficient amount of data available, thus the running time will even increase more. Another issue shown in the test of Section 4.1 is that when the number of equally partitioned disjoint subset is very high, then the number of instances within each subset will be too few to learn a model. This is the reason why we only perform the randomly partition in Section 4.3.

The performance of the Tverberg point is not shown smoothly with the increasing number of features of the dataset and the fixed number of single models in Section 4.1. One of the reason should be that the performances of single models are not guaranteed, thus the performance of the combined Tverberg point is also not guaranteed, which can be seen from the comparison with the test in Section 4.3. In Section 4.3, the test shows that with the increasing performances of the single models, the performance of the combined Tverberg point is also increasing.

From Section 4.2, we see that the performance of the combined Tverberg point is not increasing with the increasing number of single models. However,



(a) Errors w.r.t 30 experiments.

(b) Errors w.r.t 30 experiments.

(c) Errors w.r.t 30 experiments.

(d) Errors w.r.t 30 experiments.

(e) Running time w.r.t 30 experiments.

Fig. 6: Varying number of instances for single model test.

we can see that the running time is increasing quickly with the increasing number of single models computed. In Section 4.3, we see that with the increasing number of instances of single model learning, the running time is only performing small changes. Comparing the tests in Section 4.2 and 4.3, we could draw the conclusion that increasing the number of instances for single model learning is a more efficient way to increasing the performance of the combined Tverberg point than increasing the number of single models.

5 Summary

In this lab, the Iterated Tverberg point algorithm which was used for computing an approximate of center point is used for solving machine learning problem. The Iterated Tverberg point algorithm is used to combine a set of weak models to a much better model which can be seen from Section 4. There are also some comparisons included in Section 4, and we see that it is not proper to use Iterated Tverberg point algorithm if the dataset has many features. We also see that increasing the number of instances for each single model learning is an efficient way to increase the performance of the combined Tverberg point.

There are still some improvements could be achieved, and the first would be increasing the number of experiments in each dimension test in Section 4.1, so the result could be more reliable. However, we are not able to perform it because of the limitation of time. This also applies to the improvement of the algorithm implementation mentioned in [Miller and Sheehy, 2008], which says using Tverberg r-partitions instead of Radon partitions.

Bibliography

Magic gamma telescope data set. `https://archive.ics.uci.edu/ml/datasets/MAGIC+Gamma+Telescope`, May, 2007. Accessed: 8-August-2015.

G. L. Miller and D. Sheehy. Approximate center points with proofs, 2008.

T. Nguyen. Algebraic methods in combinatorics. `http://www.math.cornell.edu/~eranevo/homepage/ConvNote.pdf`, Sep, 2008. [Online; accessed 31-July-2015].

D. Sheehy. Computational geometry, lecture 17,18. `http://www.cs.cmu.edu/afs/cs/academic/class/15456-s10/ClassNotes/`, March, 2010. [Online; accessed 31-July-2015].