FACULTAT D'INFORMÀTICA DE BARCELONA

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

MASTER IN INNOVATION AND RESEARCH IN INFORMATICS

DATA SCIENCE SPECIALTY

From a Topological approach for Classification to kNN

Simon Baltes

supervised by

Lluís A. Belanche Muñoz

Dept. of Computer Science - UPC

Date of Defense:

October 29, 2020

1 Abstract

This work proposes a possible generalization of k-nearest neighbors rule (kNN) where the impact of a point goes beyond the exact location. The decision criteria does not only depend on the distances to points but also on the distance of those to the nearest miss. This concept is defined in this work as a second order interaction and it could be extended to high orders. The motivation of this concept is to take into consideration if points are in pure areas or if there are points of other classes nearby. Indeed, the methodology used to integrate the concept of second order interaction in the model is via hyper spheres centered around the points in the dataset. Those spheres have a radius that is proportional to the nearest miss and their unions and intersections will drive the decision rule. Therefore, we call this family of models Hyper Sphere Wrappers (HSW).

Using those concepts, we developed three versions of the model: One-Class HSW for binary classification, Multi-Class HSW for multi-label classification and Complete-HSW for regularized multi-label classification. Indeed, we show that the latter converge asymptotically to a local kNN rule and in particular cases to the classical kNN rule. Once the models are defined, we apply them to diverse benchmark data sets and compare them with typical machine learning models. The results are promising, in particular for high dimensional data. In addition, the complexity of the model enhance those kind of datasets because, as most distance based method, HSW scale better with the number of dimensions than with the number of samples.

Moreover, the work shows that even if the Minkovsky metric for $p < 1$ is no longer a proper distance measure, it help the algorithm to gain accuracy in high dimensional data. The results indicate that in those cases the model perform a kind of instance selection procedure. The idea is that the model will select as nearest neighbors those points that lie parallel to any local axis of the sample that we want to predict. In consequence, the model obviate points with variability in lots of different directions which seem to help to boost performance of HSW and kNN.

Finally, the work also shows that metric learning algorithms like Neighborhood Component Analysis (NCA) are able to increase substantially the performance of HSW. However, those kind of algorithms scale poorly in high dimensional data. Therefore, we leave a set of open questions that could be interesting to enhance this kind of rules. Among others, the study of Kth order interactions, the impact of defining the radius of the spheres as non-linear functions of the nearest miss or a metric learning algorithm that is able to optimize p instead of the linear transformation.

Contents

2 Introduction

K-Nearest Neighbors is probably one of the most studied non parametric model due to its simplicity but yet powerful characteristics. This algorithm was discovered in the seminal works of Fix and Hodges [1] back in 1951 and later published by Cover and Hart [2]. The k-nearest neighbors rule assign a given instance to the majority class of the k-nearest points in the training set. Diverse variants of this algorithm has been proposed in the past years [3]. Each of them addressing one of the main points of interest of the algorithm. Among others, the theoretical derivation of the optimal weights in kNN [4], the optimization procedure to find the linear transformation that optimize the cross validation score in kNN [5] or the ball tree structure to speed up the nearest neighbor search [6].

In this work, we propose a distance based classification rule inspired by topological concepts like the open balls. A topology is just a family of subsets that are called open sets which are closed under finite intersections and unions. When this open sets are chosen to be open balls, then we have the open ball topology. Those open balls with arbitrary center c and radius r are defined as the set of points x satisfying $d(x, c) < r$ [7]. The motivation of this work is to use the key property of the topology base, we can create any open set with the intersection or union of open balls. This property will allow us to shape the decision boundaries of the classifier with open balls centered in the points of the data set. In addition, this work shows that by selecting the radius of the open balls as a function of the nearest miss we can create an accurate classifier. The notion of using spheres to shape the decision boundaries have been used already in diverse studies. For instance, one class SVM [8] use a spherical large margin as data description or large margin nearest neighbor classification (LMNN) [9] that search for optimal linear transformations that compress samples of equal class into spherical regions. In addition, there are gravitational methods like GFRNN that use fixed radius NN to select the set of candidates to apply the sum of gravitational pattern [10].

The initial topological motivation lead to a general problem setting in binary classification. Given a set of points that belong to one of two classes, finds a set of N open balls with centroids c_i and spheres r_i that better separate both classes. The border arising from the intersection and union of those balls will define the decision boundary of the classifier. The exposed setting can have multiple solutions which can be arbitrary complex depending on the number of parameters that we want to optimize. Due to the problem definition we call the this family of algorithms Hyper Sphere Wrappers (HSW). Indeed, we generalize this idea to any geometric shape that arise from the L-p norm [11]. As mentioned, we use the information about the nearest miss to give a solution to the problem. This quantity has been widely used in feature selection algorithm like Relie-F [12]. Principally, this work focus around the linear dependence of the radius with the nearest miss but apriori many other dependencies could be assumed.

The advantage of working with this open balls instead of points is that we can take into consideration second order interactions. With second order interactions we mean that the decision rule not only depends on the distances to points but also on the distance of those with it's nearest miss. In consequence, if the distance to a point is big but at the same time this point is very far away from his nearest miss, then the effective distance is small. This setting appear convenient for classification since it somehow encodes the information about the class certainty of a point. In other words, classifying a point as class A because it is very close to another point of class A seem to be more solid if this point is at the same time very far away from any other point of class B.

Finally, regarding the structure of the report: Section 3 expose the trivial solution to the proposed setting from which we derive the one class model which we expand to multi-class problems. In section 4, we will add regularization parameters that define the complete model and we show that those parameters will allow to link with kNN in the asymptotic case. A part, we will also study how metric learning algorithms impact the performance of the algorithm. Finally in section 5, we expose the main conclusions and possible related future works.

3 HSW Rule

The main concept of the classification rule is to use the intersection and unions of hyper spheres to shape its decision boundaries. We will start by explaining the One-Class HSW where only one class expand spheres around the points and afterwards we generalize the idea for multi-label classification.

3.1 One-Class HSW

3.1.1 Definition

The one sided version is the simplest of its kind and it only considers the binary class setting. The problem definition is as follows:

> Given a set of N data points that belong to one of both classes S $\in \{S^-, S^+\}$, an interior class $S_c \in \{S^-, S^+\}$ and a p-norm. Find a set of M spheres such that all points of the class S_c lie in the interior of the spheres and the points of the opposite class in the exterior. The solution will be defined by the set of centroids A_i and radiuses R_i $\forall i \in [1, M]$ that better separate the both classes.

The One-Class HSW is build around the trivial solution: We choose one of the classes S as interior and each of its corresponding data points will represent a centroid of a sphere with radius equal to the nearest miss. This is indeed a valid solution that perfectly separates both classes. In figure 1, we show a 2D example of the classifier for different choices if the interior class (S_c).

The implementation is straightforward since the train procedure only need to calculate the distance matrix and the nearest miss for each point in the interior class. Afterwards, the predict procedure check whether the point that we want to classify lie inside of any sphere. If it does, then the point is classified as S_c and otherwise as S_{nc}

Algorithm 1 Train One-Class HSW

```
Input: Set of indices I over the train data rows, Interior class $S_c$ and a p-Norm
Output: Distance matrix D and the radius of each sphere R
  $D \leftarrow$ Distance Matrix between Train Points for the choosen p
  $\mathcal{I}_c \leftarrow$ Set of Points s.t. class(I) = $S_c$
  $\mathcal{I}_{nc} \leftarrow$ Set of Points s.t. class(I) != $S_c$
  $R \leftarrow \emptyset$ {Initialize solution}
  for all $i \in \mathcal{I}_c$ do
    $R \leftarrow R \cup min(D(i, \mathcal{I}_{nc}))$ {Append the nearest miss to the array of radius R}
  end for
  return $R, D$
```

Algorithm 2 Predict One-Class HSW

```
Input: Set of indices I over the test data rows, Set of radius R, Interior class $S_c$
    and a p-Norm
Output: Predicted class for each Test Point
  $S_{nc} \leftarrow$ Exterior class. $S_{nc} = S \setminus \{S_c\}$
  $Pred \leftarrow \emptyset$ {Initialize solution}
  $Dtest \leftarrow$ Distance Matrix between Test Points and Spheres
  for all $i \in \mathcal{I}$ do
    $Outside \leftarrow Dtest[i,] \geq R$ {Array of booleans indicating if the point lie outside each
    sphere}
    if $sum(Outside) = |R|$ then
      $Pred \leftarrow Pred \cup S_{nc}$
    else
      $Pred \leftarrow Pred \cup S_c$
    end if
  end for
  return Pred
```

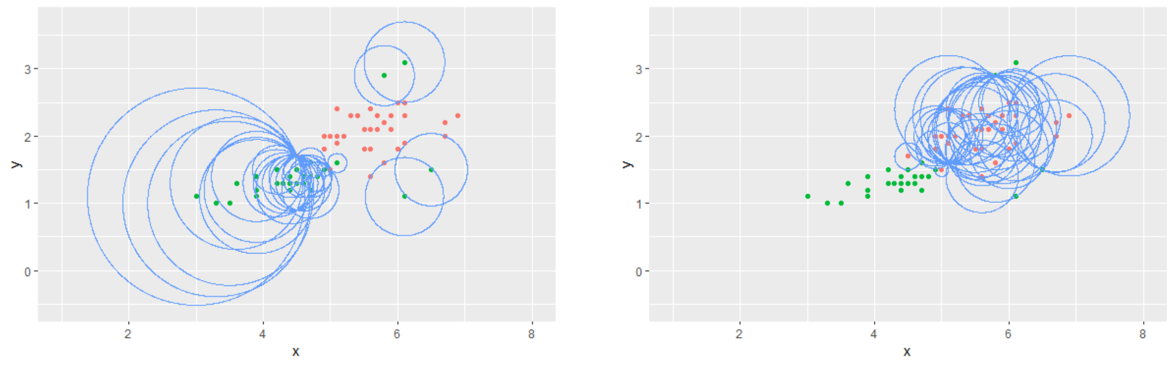

Figure (1) One-Class HS Comp decision boundaries for different choices of the interior class

As most distance based algorithms, the computation time scale better with the number of dimensions than with the number of data points. In particular, for the train procedure, the distance matrix computation scales as $\mathcal{O}(N^2P)$ and the for loop as $\mathcal{O}(N_c)$. The dependence on the dimensionality in the distance matrix computation appear because each pair of points involves a dot product of two vectors of length P.

According to this characteristics, we can already anticipate that large data-sets will be prohibited on this kind of algorithms. However, we will explore the capability of this algorithm in small sample size or even high dimensional environments.

3.1.2 Hyperparameters, symmetry and compactness

Currently the model has two hyper parameters that need to be adjusted in order to ensure the best possible generalization error.

- Interior Class S_c. This value determines which class will be associated with interior region of the hyper-spheres.
- Shape of the spheres p. This is the value that determines the minkovsky metric and therefore the norm. For instance p = 2 correspond to the euclidean distance and p = 1 to the Manhattan distance.

Both parameters are useful to balance variance and bias but the interior class is by far the one that has the strongest impact. In this section, we will investigate which characteristics/structures of a given class can cause a good generalization error and which ones not.

Hypothesis I: The class with the highest spherical symmetry should be the interior class.
Reasoning: Imagine you have a class with very little spherical symmetry and you expand spheres around them. Due to the lack of symmetry the optimal radius in both directions should be very different. In consequence, we would end up covering a region with very low density of points in one direction and therefore induce potential for miss-classifications

In order to measure the spherical symmetry of each class we created an index that captures directional heterogeneity. The main concept is to calculate the radius that a semi-sphere in one direction should have to enclose 95 % of the points an we compare it to the radius calculated in the opposite direction

In Figure 2, we show the graphical representation of this index and in practice we do the calculation as follows:

1. Select a random point in the dataset.
2. Consider all points above the semi-plane perpendicular to the maximum distance (R_{max}) and calculate the radius (R_a) such that 95 % of the points lie withing the semisphere.
3. Considering the points below the semi-plane (opposite direction than R_{max}), calculate the radius (R_b) such that 95 % of the points lie withing the semisphere.
4. The value SI = $\frac{min(R_a,R_b)}{max(R_a,R_b)}$ is what we define as the symmetry index.
5. Repeat for other random points of this class and return the average SI.

Note that if the value SI is close to 1 it means that that 95 % quantile radius is similar in both directions and therefore the class has a high level of symmetry. In contrast, small levels of SI indicate a asymmetric shape like in Figure 2.

Hypothesis II: The class with less variance should be the interior class.
Reasoning: Since we have a finite number of points, the interior region will be always cover less volume than the exterior region. Therefore, it seems to be less risky to have the high variance class outside.

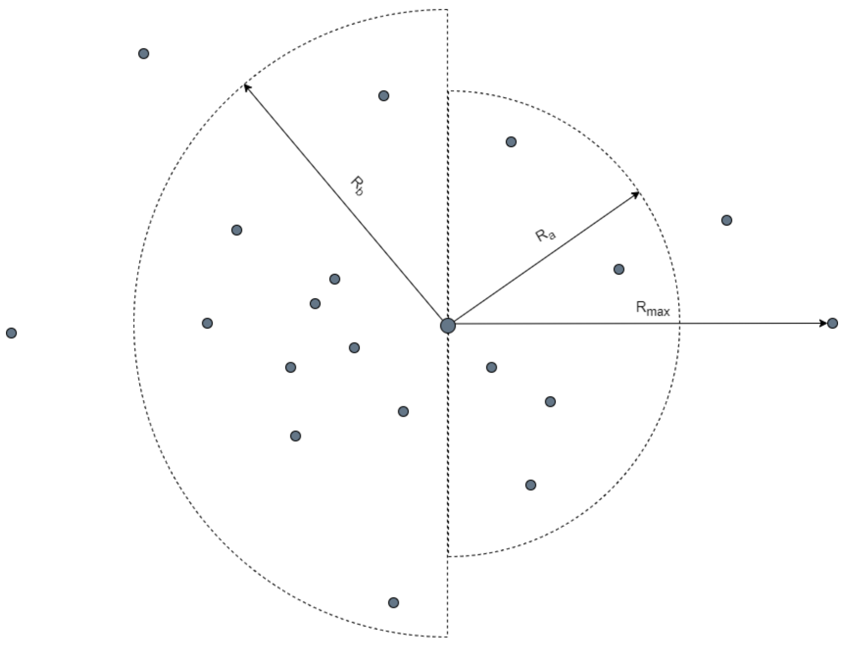

Figure (2) Symmetry Index based on the 95 % quantile in each direction.

Table 1 shows the results of the symmetry and compactness index on diverse datasets. The complete information about each dataset can be found in table 2. Regarding the methodology, we calculated cross validation classification accuracy for different values of the interior class. Based on those results, column *Best Class* indicate which had smaller error. Ideally, the best interior class in terms of CV error should be the one with highest SI and lowest CI (standardized variance). Bold fonts indicate that there is a criteria match. For instance in the first row, both SI and CI index predict correctly the best class.

Symmetry and Compactness Index					
Dataset	SI Class I	SI Class II	CI Class I	CI Class II	Best Class
Appendicitis	**0.80**	0.65	**0.13**	0.19	**Class I**
Banknote Authentication	0.51	0.44	2.03	3.15	Class II
Diabetes	0.41	0.43	**62.77**	83.34	**Class I**
Ionosphere	0.91	0.58	0.29	0.67	Class II
Qsar	0.77	0.61	6.62	**2.74**	**Class II**
Quickbird	**0.85**	0.61	85.00	47	**Class I**
Sonar	0.43	0.75	**0.12**	0.14	**Class I**
Wisconsin Breast Cancer	0.40	**0.60**	0.92	**0.76**	**Class II**

Table (1) This table shows the symmetry (SI) and compactness (CI) index for each class in the corresponding dataframe. The rightmost column indicate the interior class that show lower 10 fold CV error. Ideally, the winning class should be the one with highest SI and lowest CI.

Unfortunately, it turns out that neither of both indicators predict accurate enough the interior class on real data sets. The interior class is very important to ensure a solid generalization error and therefore we cannot rely on those indicators. Even thought, the symmetry index could be useful in other fields like kernel selection for SVM. For instance, RBF kernels give good results on datasets where the class location has a radial dependency. Therefore, we could anticipate that a RBF kernel could be worth to try on a dataset with high spherical symmetry index.

However, for this work we need to rely on cross validation to select the hyper-parameters. Even thought, most of the times hyper-parameters p and S_c have little interaction. In other words, if selecting Class X as interior show significantly higher CV error for a given p, it will hold for any value of p. In contrary, if the difference is small we have to include all pairs (S_c, p) in the grid search for the optimal hyper parameters.

A good practice to save computation time could be:

1. Run the model for a given p and check the CV error with each class.
2. If the CV error difference is bigger than 5 accuracy points:
   (a) Fix the winning class and do a line search to optimize p
   (b) Run the model with the optimal p and double check that the selected interior class is still the best choice.
3. If the CV error difference is smaller than 5 accuracy points:
   (a) Do the full grid search for each value of (p,S_c)

3.2 Multi-Class HSW

3.2.1 Definition

One-Class HSW has two main drawbacks: the interior class hyper-parameter has a very high impact on the classification error and the one sided version is only applicable to binary classification settings. Therefore, in this section, we propose the Multi-Class HSW variant of the model that does not require the definition of an interior class and is applicable to any number of labels C.

The Multi-Class variant is similar to the one sided version with respect to the training procedure but it has some key changes in the predict function. The only difference in the train procedure is that we expand the spheres around all training points (all classes) instead of only the points corresponding to the interior class. It is true that this ads additional complexity but at the same time we remove one hyper-parameter. Indeed, if we look at the whole process of training + hyper-parameter search, the complexity is similar. In the one sided version the loop is smaller because we only have to traverse the points of the interior class but we had to train the model C times to find the best performing interior class. In the multi-class version of the algorithm, we have to traverse all data points but we only have to train it once. Indeed, the latter is slightly faster because we do not need to predict in order to check the performance of each interior class.

In contrary, the Multi-Class HSW prediction rule is no longer straightforward. In the one sided version, the boundaries of the intersecting spheres separate one class from another whereas now we can find points that are within multiple spheres of different classes simultaneously. In addition, it can also happen that a test point is not contained in the interior of any sphere. From now on, we will assume binary classification setting (C = 2) in order to show how we deal with those situations.

Consider a trained Multi-class HSW model composed by a set of training points, 2 classes and a sphere placed in each point with radius equal to the corresponding nearest miss. Given a new data point, it will face one of those two scenarios:

- **CASE 1**: The new training point lie in the interior of at least one sphere. In that case, we treat the situation as a voting system (simple majority vote). For instance, if the new data point lie within 80 % of the C_1 spheres but only within 40 % of the C_2 spheres we will predict this point as C_1. In case of tie, we choose one class at random.

- **CASE 2**: The new training point does not lie in the interior of any sphere. In that case, we associate the new point to the class of the nearest sphere. Note that nearest sphere is defined as the nearest border, not the nearest centroid.

In figure 3, we show an example for each case and in the next page we describe the pseudocode for Multi-Class HSW.

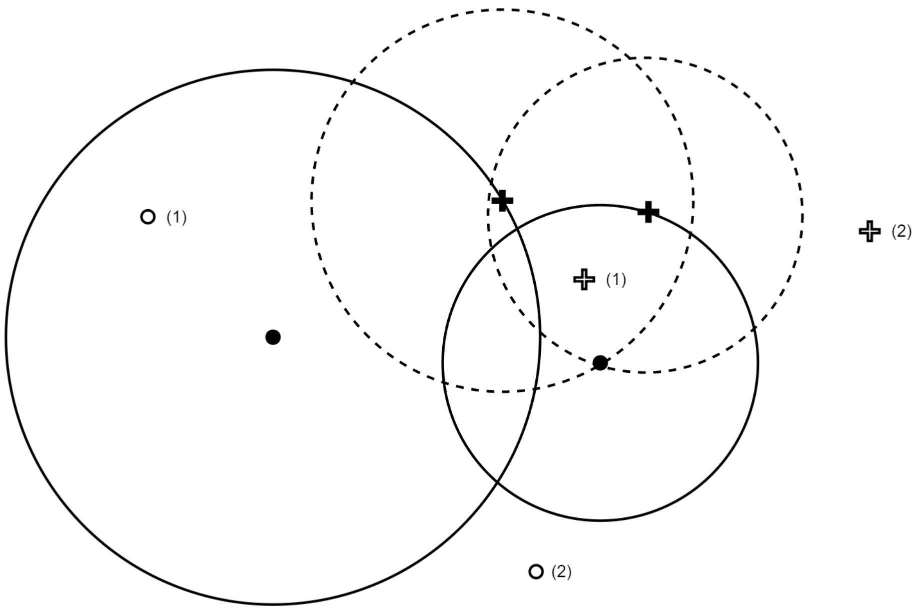

Figure (3) Graphical representation of the Multi-Class HSW for C = 2. Two classes (cross and circles), train points (bold shapes), test points (empty shapes) and fitted spheres (continuous spheres indicate circle spheres and discontinuous spheres indicate cross spheres). The number in the parenthesis indicate to which of the previously explained cases the test point correspond.

Algorithm 3 Train Multi-Class HSW

```
Input: Set of indices I over the train rows, Set of classes $C_j$ of the target variable and a norm p
Output: Distance matrix D and the radius of each sphere R
  $D \leftarrow$ Distance Matrix between Train Points for the choosen p
  $\mathcal{I}_{C_j} \leftarrow$ Set of Points s.t. class(I) = $C_j$
  $R \leftarrow \emptyset$ {Initialize solution}
  for all $i \in \mathcal{I}$ do
    for all $C_j \in \mathcal{C}$ do
      if class($i$) = $C_j$ then
        $R \leftarrow R \cup min(D(i, \mathcal{I}_k)) \quad s.t \quad k \neq C_j$
        break
      end if
    end for
  end for
  return $R, D$
```

Algorithm 4 Predict Multi-Class HSW

```
Input: Set of indices I over the test data rows, Set of fitted spheres R and a value for p (Norm)
Output: Predicted class for each Test Point
  $Pred \leftarrow \emptyset$ {Initialize solution}
  $Dtest \leftarrow$ Distance Matrix between Test Points and Spheres
  for all $i \in \mathcal{I}$ do
    $Outside \leftarrow Dtest[i,] \geq R$ {Array of booleans indicating if the point lie outside each sphere}
    if $sum(Outside) = |R|$ then
      $Pred \leftarrow Pred \cup NearestSphere$
    else
      $Pred \leftarrow Pred \cup MajorityVote$
    end if
  end for
  return $Pred$
```

Algorithm 5 NearestSphere

```
Input: The ith data point, Array of distances from i to train set Dtest[i,] and Set of fitted Spheres R
Output: Predicted class for the corresponding point based on the nearest sphere criteria
  $Pred \leftarrow \emptyset$ {Initialize solution}
  $MinDist \leftarrow \infty$ {Initialize Minimum Distance}
  for all $R_j \in \mathcal{R}$ do
    if $D[i,j] - R_j < MinDist$ then
      $MinDist \leftarrow D[i,j] - R_j$
      $Pred \leftarrow Class(R_j)$
    end if
  end for
  return $Pred$
```

Algorithm 6 MajorityVote

```
Input: The ith data point, Array of distances from i to train set Dtest[i,] and Set of fitted Spheres R
Output: Predicted class for the corresponding point based on the majority vote
  $Pred \leftarrow \emptyset$ {Initialize solution}
  $Votes_{C_j} \leftarrow 0$ {Initialize the number of votes for each class $C_j$}
  $Spheres_{C_j} \leftarrow 0$ {Number of spheres/train samples of class $C_j$}
  for all $R_j \in \mathcal{R}$ do
    if $D[i,j] < R_j$ then
      $C_j \leftarrow \text{Class}(R_j)$
      $Votes_{C_j} \leftarrow Votes_{C_j} + \frac{1}{Spheres_{C_j}}$
    end if
  end for
  $w \leftarrow argmax_j(Votes_{C_j})$
  if $CountDistinct(w) = 1$ then
    $Pred \leftarrow C_w$
  else
    $Pred \leftarrow$ Select a Random Class between all $C_w$
  end if
  return $Pred$
```

3.3 Empirical Results

3.3.1 Experimental Design

We dedicate this small section to explain how we calculated the results for section 3.3.2.

First at all, the objective of 3.3.2 is to compare One-Class and Multi-Class HSW with other models. The decision of which models to compare is somehow arbitrary but we tried to pick them as diverse as possible. First, a purely distance based rule with low complexity like kNN. Then, a linear classifier like SVM that tend to give good results on higher dimensional data and is robust to over-fitting. And third, a gradient boosting model that include tree structures like XGboost. In order to do a fair comparison, we did not consider to use kernel kNN or kernel SVM. It would be interesting to compare kernelized versions if we apply them to both HSW and SVM. Note that HSW can be kernelized since it is a distance based method.

Regarding the dataset, the decision was made based on popularity such that they can be considered as benchmark datasets for classification. A part from the results that we show in table 3, those datasets have been widely applied in literature such that it help us to evaluate the results of HSW.

The methodology used to calculate the classification error in table 3 is as follows:

- Split into train and test set. The fraction used for training is 2/3 and for testing 1/3.
- Hyperparameter tuning with 10 fold cross validation. When we applied cross validation on the train set, we did an exhaustive grid search for all models except XGboost due to the big amount of free parameters. For the latter we applied randomized grid search in order to speed up computation time.
- Classification Accuracy to evaluate the performance. With the set of hyperparameters that showed best CV accuracy, we train again the model on the full train set and calculate the classification accuracy on the test set.
- Repeat this procedure for different splits train/test. In order to get an idea of the robustness of the result, we repeat this procedure for different pairs of train and test sets. In particular, the procedure has been repeated 20 times and with those values we calculate the average prediction and the 95 % confidence interval. The expression used to calculate the confidence interval involve the t-statistic and thus assume unknown variance and normality over the distribution of the mean.

Therefore, the results in table 3 correspond to the mean classification accuracy and the corresponding confidence interval on the test set. The utility of the confidence interval is to assess if results are significantly different. For instance, if the mean accuracy plus minus the confidence interval of model A overlap with model B, then we cannot say with 95 % confidence that model A perform better than B on this dataset. In table 3, the mean value that better performed is marked in bold. However, if this very same model perform statistically significant better than the second placed model, then the confidence interval is also marked in bold.

Finally in table 2, we give brief description of each dataset used in table 3.

Data Description			
Dataset	Sample Size	Dimensions	Number of Classes
Appendicitis	106	7	2
Banknote	1372	4	2
Diabetes	768	8	2
Ionosphere	351	34	2
Qsar	1055	41	2
Quickbird	4839	5	2
Sonar	209	60	2
Wisc. Breast Cancer	699	10	2

Table (2) Number of samples, dimensions and classes for all datasets used in table 3

3.3.2 Binary Classification

In this section we will apply One-Class HSW and Multi-Class HSW to diverse binary classification benchmark data sets.

Application on real data and comparison					
Dataset	1-Class HSW	2-Class HSW	Lin. SVM	KNN	XGBoost
Appendicitis	84.42 ± 2.42	86.00 ± 2.55	86.42 ± 2.40	**87.57** ± 1.94	86.43 ± 2.79
Banknote	98.01 ± 0.60	99.72 ± 0.20	99.03 ± 0.18	**99.97±0.03**	99.22 ± 0.30
Diabetes	70.01 ± 1.11	72.28 ± 0.74	**77.03** ± 0.73	73.04 ± 1.29	74.92 ± 1.79
Ionosphere	**93.58** ± 0.85	**93.41** ± 1.01	88.39 ± 1.48	85.42 ± 1.66	91.97 ± 1.15
Qsar	80.54 ± 1.28	84.60 ± 0.70	**86.43** ± 1.06	83.94 ± 0.88	86.10 ± 0.67
Quickbird	95.79 ± 0.12	97.47 ± 0.18	96.78 ± 0.23	96.81 ± 0.19	**98.51±0.15**
Sonar	72.31 ± 3.86	79.76 ± 1.91	75.21 ± 2.98	**83.40** ± 1.72	81.37 ± 1.89
Wisc. Breast Cancer	96.32 ± 0.54	**97.31** ± 0.76	96.49 ± 0.34	96.65 ± 0.53	95.44 ± 1.22

Table (3) One Class HSW and Multi Class HSW comparison with other typical benchmark models.

The results in table 1 have been extracted by fitting S_c and p via 10 fold CV for optimizing One-Class HSW. In Multi-Class (2-class) HSW we used 10 fold CV to find optimal p. Finally, we fitted C value for SVM, k for KNN, and (eta, $gamma$, $maxdepth$, $minimumchildweight$, $nrounds$) for XGboost.

First, note that Multi-Class HSW performance is always better than One-Class HSW. However, in general the results show little significance in the sense that there are only two cases where we have statistical guarantees: kNN outperform in Banknote Auth. and XG-boost in Quickbird. However, Multi-Class HSW show good results compared to standard algorithms in Ionosphere and WSBC but with the current resampling size we cannot give a 95 % significance. On the other hand, on data sets like Diabetes or Qsar the performance of HSW is low whereas when the number of data points grow, boosting algorithms like XGBoost gain performance (Qsar, Quickbird).

After checking performance of other authors in the Open ML repository, we discovered that the RBF kernel applied to a SVM show similar results than HSW on Ionosphere dataset. A possible explanation is that both approaches exploit the spherical symmetry of the classes. Indeed, in table 1, we observe that ionosphere is the dataset that show highest spherical symmetry index. In addition, even if XGboost outperform in Quickbird, we observe a similar phenomena. Again, if we check table 1, we see that this dataset has a very high symmetry and at the same time HSW is significantly better than SVM or kNN. On the other hand, the performance of HSW is poor in datasets with low spherical symmetry index like Diabetes, where a linear classifier shine. Therefore, a first insight could be that it is worth to try HSW where we suspect that there is a radial dependency of the class positions.

Finally, Sonar, Ionosphere and Qsar have a relatively high dimension compared to it's sample size. The results show that this version of HSW is not yet ready to compete in Sonar or Qsar. This fact encourage us to add regularization parameters that could boost HSW in higher dimensional data.

4 Complete HSW: Regularization and Convergence towards kNN

Currently Multi-Class HSW rule has only one hyper-parameter, the value of p that we decide to use for the distance matrix calculation. In this section, we propose two additional regularization parameters that define the Complete HSW rule. In addition, those parameters will allow us to link with K-nearest Neighbors in the asymptotic case.

First, we want to give more flexibility to the Nearest Sphere selection. Remember that when a new data point is not contained in any sphere we assign the value of the nearest sphere. Thus, just as in KNN, we want to allow the algorithm to choose between the K-Nearest Spheres instead only the nearest one.

The second parameter is related to the intensity of the second order interaction. In particular, we will replace the nearest miss $R_i \to \lambda R_i$ where $\lambda \in [0, \infty)$ This parameter allow us to control the percentage of cases that fall in each setting explained in section 2.2.1. For instance, if lambda is very large then all radius will be very large and thus most points will be contained in some sphere whereas if lambda is small most of the points will lie outside the spheres. Indeed, note that in the case $\lambda = 0$ all spheres' radius are zero such that spheres get reduced to single points. In that case all the test points will lie outside the spheres and therefore we will apply the K-Nearest Sphere approach, which will be exactly the same as the KNN algorithm. On the other hand, in the limit $\lambda \to \infty$ we have the opposite case where all test points should lie within each an every sphere. Therefore, according to the majority vote, points will be assigned to the majority class (just as in the limit $k \to \infty$). In the upcoming sections, we will analyze in detail those statements and show some particular cases where they are not correct.

Finally, it is interesting to note that lambda is a scale parameter and that its value does not impact the ratio between spheres of different points. This fact is convenient because it mean that by changing lamda we will not change the hierarchy between nearest neighbors. In other words, if the radius of sphere 1 is two times bigger than sphere 2 radius, then this will hold for any value of lambda.

To conclude, remark that adding hyper-parameters k, λ is mathematically convenient because it allow us to link HSW with kNN in the asymptotic case. Thus, the performance of HSW should be always greater or equal to kNN provided that we include the value $\lambda = 0$ in the hyper-parameter search. The magnitude of improvement that this extension provide will depend on how well this second order interaction suit the data.

4.1 Limit $\lambda \to 0$

The formal question we have to answer is if the following limit exist for any new test point x.

$$\lim_{\lambda \to 0} HSW(k, \lambda, x) \overset{?}{=} KNN(k, x) \tag{1}$$

First, we recall the definition of open ball. Let (M,d) be a metric space with a given metric d. The open ball centered at c is denoted by $B_r(c)$

$$[B_r(c) \;=\; \{x \in M \mid d(x, c) \;<\; r\} \tag{2}$$

Where the condition of open requires to not include the boundary. This condition is key for the classifier because we expand the spheres up to the nearest miss.

Consider now a trained HSW algorithm as a set of N open balls of dimension P with radius r_i and centers c_i. The radius is defined as proportional to the nearest miss U_i of the point that lie in centroid c_i.

$$r_i = \lambda U_i \tag{3}$$

Any given test point x can be either in the interior or in the exterior of a given sphere i. Thus, we define the following vector to indicate whether a point lies inside or outside the spheres.

$$z = (z_1, z_2, ..., z_n) \tag{4}$$

$$z_i = \begin{cases} 1 & \text{if } d(x, c_i) < r_i \\ 0 & \text{otherwise} \end{cases}$$

In consequence if $\sum_i z_i = 0$ then we apply the nearest sphere approach and if $\sum_i z_i > 0$ we use the majority vote of the spheres i such that $z_i = 1$. In order to ease notation we will refer from now on to the summation over all z_i as $S(x)$ such that $S(x) = \sum_i z_i$

Those two cases cover all possible situations and they are disjoint in the sense that a test point can either have $S(x) = 0$ or $S(x) > 0$. Therefore we can define HSW rule as sum over both cases.

$$\lim_{\lambda \to 0} HSW(k, \lambda, x) = \lim_{\lambda \to 0} \left(\mathbb{1}_{S(x)=0} KNS(x, k, r) + \mathbb{1}_{S(x)>0} MV(x, r) \right) \tag{5}$$

where $\mathbb{1}$ represent the indicator function. Now, we can imagine that if we start to shirk lambda and in consequence the radius of the spheres, at some point the test sample x will lie outside any sphere. In a formal manner, this means that for a given point x, there should exist a value $\epsilon > 0$ such that any $\lambda \leq \epsilon$ will force the point to be outside any sphere. The fact that epsilon exists or not only depends on the distances from x to the centroids c_i and its radius r_i.

Finally, it is convenient to define j as the index of the sphere that fulfill the following condition.

$$d(x, c_j) - r_j = 0 \mid d(x, c_i) - r_i \geq 0 \ \forall i \neq j \tag{6}$$

Note that j correspond to the last sphere whose border lie exactly on the point x such that all the other spheres no longer contain it. We can now define ϵ as the value of lambda that leave the point x exactly on the border of the deepest sphere j.

$$d(x, c_j) - r_j = 0 \to d(x, c_j) - \epsilon U_j = 0 \to \epsilon = \frac{d(x, c_j)}{U_j} \tag{7}$$

From equation 7 we can derive that in order to exist $\epsilon > 0$, no repeated points such that $d(x, c_j) = 0$ should exist. Another conflict can arise when $U_j = 0$ (zero nearest miss distance). This situation can only exist when we have two points that have the exact same explanatory variables but different target class. In that case both spheres will annihilate each other in the sense that both will have radius equal to zero. In consequence, ϵ diverge but the condition $\epsilon > 0$ is still satisfied such that the only possible conflict can arise from repeated points. With repeated points we mean test points that lie exactly on the centroid of a trained sphere and in consequence $d(x, c_j) = 0$.

4.1.1 No repeated points: ϵ exist.

This special case of data sets require that we can have no test points that lie exactly on the centroid of a trained sphere. Therefore, we can assure that epsilon exist and in consequence the limit can be simplified:

$$\lim_{\lambda \to 0} HSW(k, \lambda, x) = \lim_{\lambda \to 0} KNS(x, k, r) \quad \forall\, \lambda \leq \epsilon \,|\, \epsilon > 0 \tag{8}$$

Note that now the only difference with KNN is that KNS calculate the distance to the border of the spheres instead to the point itself. Therefore we just have to prove that when the radius tend to 0, we end up having the KNN algorithm. In other words we want to show that the distance criteria in KNS converge to the distance criteria in KNN.

The distance criteria in KNS is just the distance to the border of the spheres and in KNN the distance to the points.

$$d_{KNS} = d(x, c_i) - r_i$$
$$d_{KNN} = d(x, c_i)$$

Since the radius is proportional to λ, the convergence is trivially satisfied.

$$\lim_{\lambda \to 0} (d_{KNS}) = \lim_{\lambda \to 0} (d(x, c_i) - r_i) = \lim_{\lambda \to 0} (d(x, c_i) - \lambda U_i) = d(x, c_i) = d_{KNN}$$

Therefore, we conclude that whenever there exist an $\epsilon > 0$, HSW rule converge to KNN in the limit $\lambda \to 0$

4.1.2 Repeated points: ϵ does not exist.

In the case that the point x lie exactly on the centroid c_j, then no matter which value of $\lambda > 0$ we choose, the point will be always inside the sphere. Therefore, the Majority Vote will be applied since only points that are outside all spheres fall into the K-Nearest Sphere case. What will happen is that for very small lambda we will have KNS on non repeated points and Majority Vote on the repeated points. Then once we make $\lambda = 0$ all those repeated points will fall abruptly outside the spheres and therefore they will fall into the KNS approach. This means that our classifier has a discontinuity in $\lambda = 0$ and therefore the limit exposed in equation 1 does not exist.

Thus, in the limit of small lambda we will have a local KNN approach. Non repeated points will have a value of k defined by the hyperparameter and the repeated points will have a value of k equal to the number of repeated points. For instance, if we have 3 points that lie on the centroid c_1 then $k_1 = 3$ whereas other centroid c_2 that have 4 repeated points will have $k_2 = 4$.

In general, we can say that HSW is a generalization of a local KNN rule and in some particular cases also a generalization of the classical KNN approach. We could force a dataset to have no repeated points by adding a small error term such that no point land on any centroid but for the moment we see no need for that. Bear in mind that you can always add $\lambda = 0$ in the grid search to recall the classic KNN rule.

4.2 Limit $\lambda \to \infty$

This limit can be solved with the same methodology used in the previous section. Now we want to prove that the limit $\lambda \to \infty$ HSW converge to a static prediction equal to the percentage of the majority class C

$$\lim_{\lambda \to \infty} HSW(k, \lambda, x) \overset{?}{=} \frac{N_C}{N} \tag{9}$$

Where N indicate the number of points and N_C the number of points corresponding to the majority class in the train set.

Key here is to show that there exist a value of λ such that the point x is contained in all the available spheres. Therefore, we define the opposite expression to eq 6., the farthest open ball (distance to the border).

$$d(x, c_l) - r_l = 0 \mid d(x, c_i) - r_i \leq 0 \;\forall i \neq l \tag{10}$$

Now we define that if we start to increase λ, the last sphere that will end up containing x is sphere l. In order words, we can define the threshold ν as the value of λ that leaves the point x exactly on the border of sphere l. In consequence, we can assure that whenever a finite ν exist, the limit in eq 5 can be simplified.

$$\lim_{\lambda \to \infty} HSW(k, \lambda, x) = \lim_{\lambda \to \infty} MV(x, r) \quad \forall\, \lambda > \nu \,|\, \nu < \infty \tag{11}$$

Note that now x is contained all the spheres and the value of the classifier will be constant in the range $\lambda \in [\nu, \infty]$. Due to the definition of MV(x,r), the output of the function will be equal to the majority class fraction and thus we close the proof in eq 9.

The only open question is if ν exist for any possible dataset. Using the analogous expression than eq 7. for the farthest sphere.

$$d(x, c_l) - \nu U_l = 0 \to \nu = \frac{d(x, c_l)}{U_l} \tag{12}$$

Since all distances are finite by definition the only possible corner case is when $U_l = 0$ and thus ν diverge. The fact that the nearest miss is zero means that we have points in the train set with equal explanatory variables but different class labels. In that case, no matter how big we make lambda, the radius of the sphere will be equal to zero. Therefore, we can conclude that the limit in equation 9 exist only when all spheres have strictly positive nearest miss.

4.3 Empirical Results

4.3.1 Experimental Design

Just as in section 3.3.1, the purpose in this one is to explain the methodology used to derive the results in section 4.3.

First of all, the procedure used to calculate the mean classification error and the confidence intervals in table 5, 6 and 7 is exactly the same as in table 3.

Regarding the intention of each table:

- Table 4: Show the number of samples, dimensions and classes of each dataset used in section 4.3.
- Table 5: Compare the three versions of HSW and evaluate if it is worth to fit all three hyperparameters
- Table 6: Show the results of HSW on high dimensional data and compare them to other models. We chose the ones that tend to give good results on high dimensional data like regularized logistic regression, SVM or kNN.
- Table 7: Apply metric learning algorithm NCA to HSW and kNN and compare it to the results obtained when we optimize p via cross validation

Data Description			
Dataset	Sample Size	Dimensions	Number of Classes
Appendicitis	106	7	2
Banknote Auth.	1372	4	2
Diabetes	768	8	2
Ionosphere	351	34	2
Qsar	1055	41	2
Quickbird	4839	5	2
Sonar	209	60	2
Wisc. Breast Cancer	699	10	2
Iris	150	4	3
Wine	178	14	3
Balance Scale	625	4	3
Omentum Kidney	337	10937	2
Breast Ovary	542	10937	2
Ovary Lung	324	10937	2
Omentum Prostate	146	10937	2
Lung Kidney	386	10937	2
RSCTC2 2010	105	22284	3

Table (4) Number of samples, dimensions and classes for all datasets used in section 4.3

In addition, it is worth mentioning the procedure used to derive results for the instance selection process (4.3.5) and the asymptotic behavior (4.3.6).

The methodology used to plot figure 7 is to calculate the pairs distance and angle with the closest axis for each nearest neighbor. More precisely:

1. For each point in the test set, we take the 100 nearest neighbors and store their distances
2. Then, for each nearest neighbor we calculate the angles to each and every axis and store the minimum value. Thus, we will have for each point in the test set and for each nearest neighbor a pair of values (distance, angle)

3. Finally, in order to reduce the sample size, we average on each nearest neighbor. For instance, take the first nearest neighbor of all test samples and average their distances and angles and repeat this process until we get the values for the 100th nearest neighbor.

Therefore, what we plot in figure 7 are the pairs of average distance vs average minimum angles of those 100 nearest neighbors for different values of p.

On the other hand, the calculation of figure 8 is more simple. This plot shows the density function over all pairs of distances from test to train samples for different values of p. In both figure 7 and 8, the leftmost plot correspond to Wisconsin Breast Cancer dataset and the right one to Ionosphere.

Regarding section 4.3.6, the objective is to show that HSW converge to kNN in the limit $\lambda \to \infty$ when we have no repeated points (test points match with centroids). The methodology used to parametrize the repetition of points is via the Oversampling factor (OF). The idea es that we pic at random a percentage of points from the original data set and repeat them so that they appear exactly two times in the data set. The selected percentage of the sample size to repeat is exactly equal to the oversampling factor. For instance, OF = 0.05, means that we select at random 5 % of the rows and duplicate them.

Finally, the dataset used in section 4.3.6 is synthetic. We create a 2D dataset that has two concentric classes arising from a Gaussian distribution over the radius. More precisely, the distance from class i to the center has a normal distribution with mean R_i and variance σ_i. In order to avoid negative values of the inner class, we take the absolute value. In figure 10, we show an example of the data set for different values of the variance.

4.3.2 Classification

This section show the performance of Complete HSW rule. We already showed some results of HSW for $\lambda = 1$ and $k = 1$ (Multi-Class HSW) in section 2.3.1 on typical benchmark datasets. Now we explore the full power of the algorithm and study the difference between Complete HSW and Multi-Class HSW. Therefore, if it is worth to fit all 3 hyper-parameters (p,lambda and k).

Table 5 show from left to right: HSW with optimal (k,λ) and p = 2, HSW with optimal p and (k = 1,λ = 1) and HSW with optimal (k,λ,p)

Different number of parameters optimized with CV			
Dataset	HSW opt.(k,λ)	HSW opt. p	HSW opt.(k,λ,p)
Appendicitis	**87.44** $\pm$ 2.60	86.00 $\pm$ 2.55	86.00 $\pm$ 2.60
Banknote Auth.	**99.98** $\pm$ 0.03	99.72 $\pm$ 0.20	99.95 $\pm$ 0.04
Diabetes	**73.35** $\pm$ 1.06	72.28 $\pm$ 0.74	72.93 $\pm$ 1.14
Ionosphere	93.29 $\pm$ 1.1	93.41 $\pm$ 1.01	**94.23** $\pm$ 1.05
Sonar	**82.75** $\pm$ 2.38	79.76 $\pm$ 1.91	82.61 $\pm$ 1.61
Wisc. Breast Cancer	96.87 $\pm$ 0.77	**97.31** $\pm$ 0.76	96.79 $\pm$ 0.81
Iris	95.20 $\pm$ 1.01	93.77 $\pm$ 1.57	**97.00** $\pm$ 1.90
Wine	96.10 $\pm$ 2.62	**97.45** $\pm$ 1.51	96.61 $\pm$ 1.68
Balance Scale	**90.62** $\pm$ 1.11	87.83 $\pm$ 1.92	90.01 $\pm$ 1.05
Omentum Kidney	94.01 $\pm$ 1.39	97.01 $\pm$ 1.22	**97.76 $\pm$ 1.03**
Breast Ovary	89.33 $\pm$ 2.11	95.11 $\pm$ 0.73	**95.72 $\pm$ 0.90**
Ovary Lung	89.53 $\pm$ 1.41	91.01 $\pm$ 1.11	**94.15 $\pm$ 1.49**
RSCTC2 2010	52.34 $\pm$ 3.40	53.28 $\pm$ 4.68	**58.28** $\pm$ 4.15

Table (5) Complete HSW with different number of free parameters

In general, we observe that on most low dimensional datasets there is little difference between the three variants. There are datasets where the value of p is more important and others where fitting 3 parameters traduce in an excess of complexity which lead to over-fitting. However, in general there are no statistically significant differences among the three settings. Therefore, on low dimensional data it seems that there is no point in doing a exhaustive grid search to optimize all three hyper parameters. There is little gain in accuracy and the computing time is heavy because we need to recalculate the distance matrix for each value of p. A randomized grid search would be more suited for this kind of data sets.

In the other hand, on most high dimensional datasets, the classification accuracy is significantly better on the three parameter setting. The biggest improvement comes from optimizing the value of p whereas fine tuning k and λ consistently add small improvements.

4.3.3 Classification on high dimensional datasets

Due to the positive results of HSW, we want to further explore the performance on high dimensional data sets. To do so, we will use datasets from the gene expression machine learning repository (GEMLeR).

DNA Microarray Sequences - Comparison					
Dataset	KNN	KNN p opt	Logistic	Complete HSW	Lin. SVM
Oment. Kidney	96.60 ± 1.64	96.88 ± 1.22	97.11 ± 0.57	**97.76** ± 0.74	97.33 ± 0.62
Breast Ovary	93.27 ± 1.18	94.44 ± 1.83	**96.90**±0.39	95.72 ± 0.90	96.54 ± 0.45
Ovary Lung	89.07 ± 2.88	91.64 ± 1.27	**94.16**±0.97	94.15 ± 1.49	94.02 ± 0.91
RSCTC2 2010	50.85 ± 3.25	52.00 ± 5.59	53.42 ± 4.65	**58.28** ± 4.15	57.61 ± 4.74
Oment. Prostate	94.58 ± 2.09	97.50 ± 1.80	97.85 ± 0.86	**98.13** ± 1.28	97.75 ± 0.66
Lung Kidney	95.46 ± 0.99	96.40 ± 0.78	96.54 ± 0.46	96.83 ± 0.86	**96.93**±0.59

Table (6) HSW performance on high dimensional data compared with other benchmark models

Table 6 compare HSW with most common used algorithms on high dimensional data. Mostly algorithms with low complexity like linear SVM , logistic regression or kNN have more chances to deal with over-fitting in this kind of data. In order to do a fair comparison of HSW with kNN, we also include the version of kNN with optimal p. Logistic regression and SVM has been trained with LiblineaR package in R. The best possible regularization method and value has been selected via 10 fold cross validation. Logistic regression showed best performance with L_1 regularization whereas in SVM, L_1 regularization and L_2 loss.

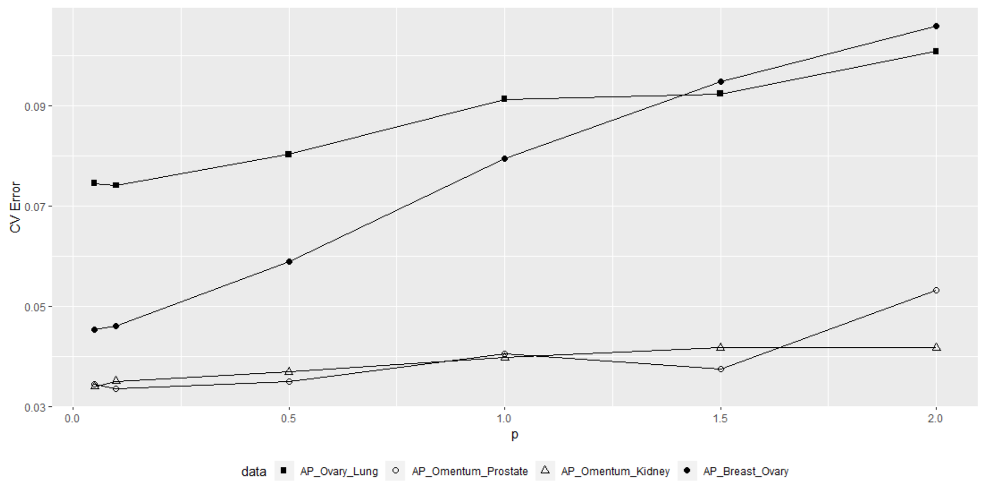

Figure (4) Cross validation error on the train set for different values of p

Regarding the results, in almost all datasets HSW perform significantly better than kNN with p = 2. Indeed, there is one out of six datasets (Ovary Lung) where HSW is also significantly better than kNN with optimal p. In contrary, when we compare HSW with Logistic Regression or SVM, there is no significant difference among them.

4.3.4 Instance Selection Process for $p \leq 1$

After applying HSW and kNN to high dimensional data, it turned out that the values of p that showed best results are far below 1. As figure 4 indicate, on GEMLeR data sets, the successful values of p where between 0.05 and 0.1. This is surprising considering that the l-p norm for $p < 1$ is no longer a proper distance metric due to the lack of the subaditivity property [13]. As you can see in Figure 5, the hypersphere for $p < 1$ have a star like appearance. The star has as two vertices per dimension and for a fixed radius, the volume get smaller and smaller as we decrease p (figure 6).

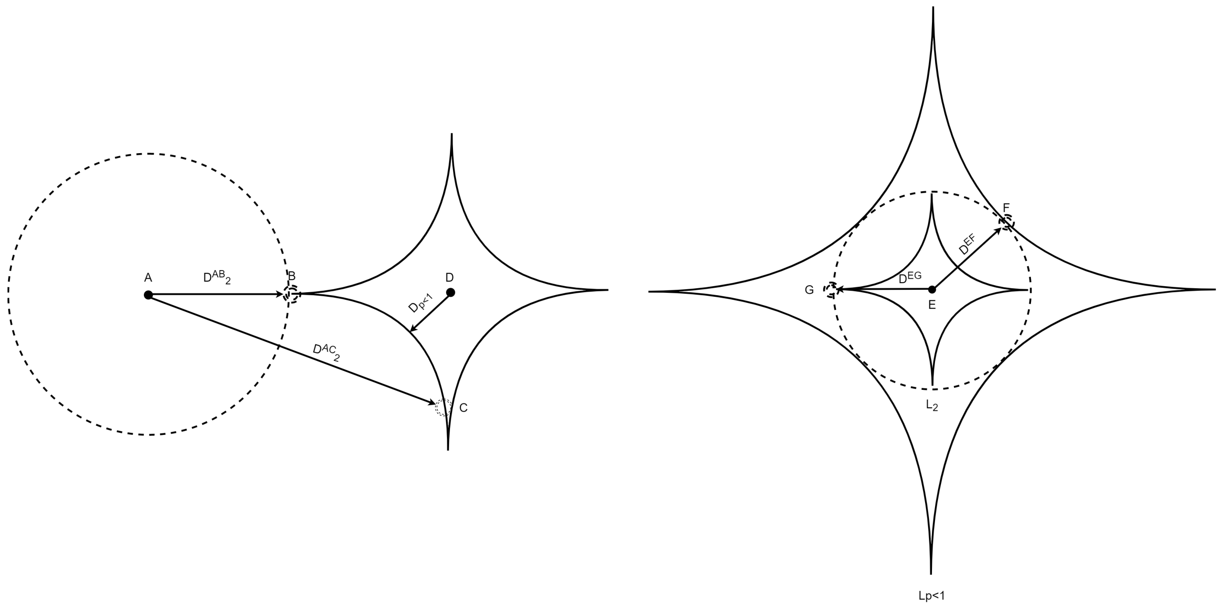

Figure (5) Graphical representation of the distance distortion along the coordinate axis for $p < 1$. The stars represent the unit sphere circle for some value $p < 1$. The discontinuous circles indicate the unit circle for the euclidean distance metric.

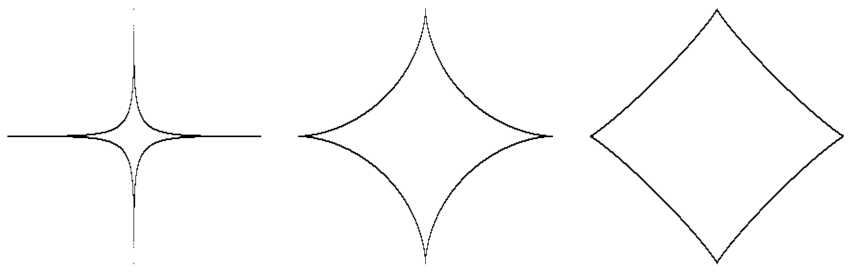

Figure (6) Unit radius circle for different values of p. From left to right p = 0.3, 0.6, 0.9

Therefore for very small values of p, only points that lie on some of the local axis [1] will be relevant for the nearest neighbors selection. All the other points will be so far away that they are excluded from the nearest neighbors selection unless we have a very high value of k. For instance in figure 5, note that point G and F are at the exact same euclidean distance from point E. However, for values $p < 1$, point G is much closer than F. Now imagine that we start to shrink more and more the value of p (figure 5) such that all the points that are not directly aligned with the axis will be so far away from E that they will be irrelevant for the algorithm. In addition, it is easy to show that the distance between two points that lie on the same local axis is independent of p.

If the previous statements are true, then the distance to the nearest neighbors will have a dependency with the angle respect to the local axis. In other words, if we decrease the value of p, kNN will select as nearest neighbors those samples that lie parallel or almost parallel to some axis (small angle). In figure 7, we plot the average distance of each nearest neighbor vs the minimum angle to any local axis. Indeed, note how the dependency with the angle increase as we lower the value of p. For instance, p = 1 the shape of the unit sphere is a rhombus such that the vertices already start to activate the instance selection process. However, this process intensifies for smaller values of p. On the other hand, note that for p = 10 the distance has absolutely no dependency with the angle to the local axis. In figure 8 we can observe how those two groups of points appear when we move to small values of p (0.01 or 0.05)

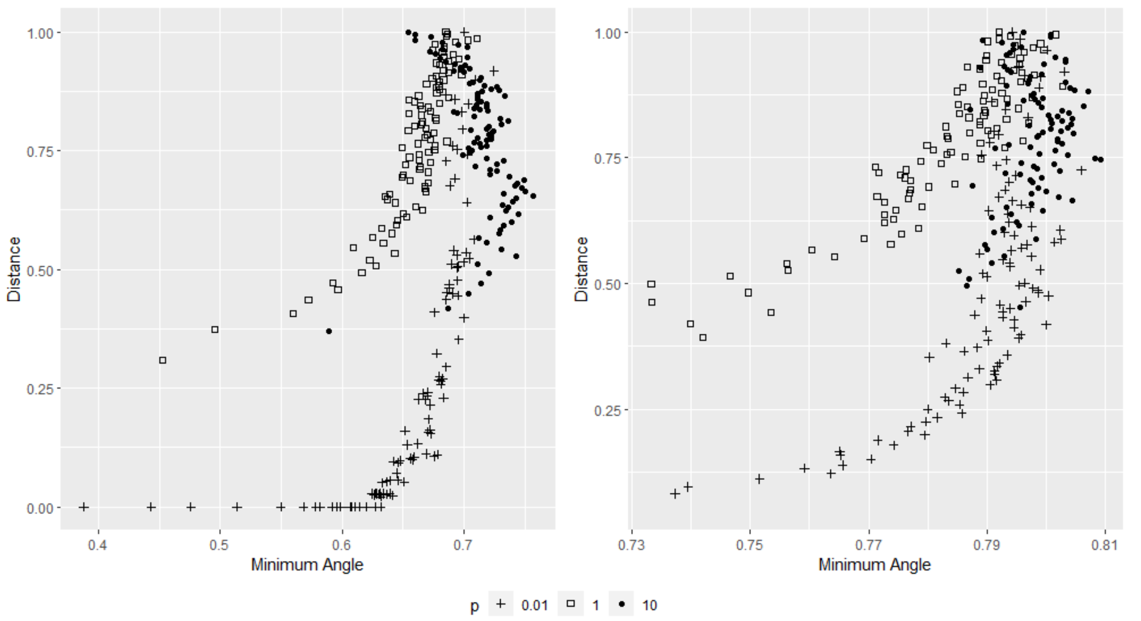

Figure (7) Distance vs angle dependency for different values of p. Left plot correspond to Wisconsin Breast Cancer and right one to Ionosphere data set.

It is yet unclear how this process suits high dimensional data but the results show that the CV decrease as we decrease the values of p. The hypothesis is that only comparing with other points along the local axis help the model to detect patters. With noise we mean points that have a variability in all directions. Intuitively, we could say that if order appears clear in the data, the algorithm might detect easier relevant patterns. For instance, it is clear that the vector (0,1) is smaller than (0,2) but the answer is not so clear when the comparison is (2,1) vs (0,4). However, those are just possible hypothesis. Truth is that but we don't know is this is a industry related effect (genomic data) of if this is an intrinsic characteristic of the nearest neighbor approach in high dimensional data. More theoretical analysis would be needed to clarify this question.

[1] Local axis: Coordinate system obtained by translating the axis to the corresponding point a. $\vec{x} \rightarrow (\vec{x} - \vec{a})$

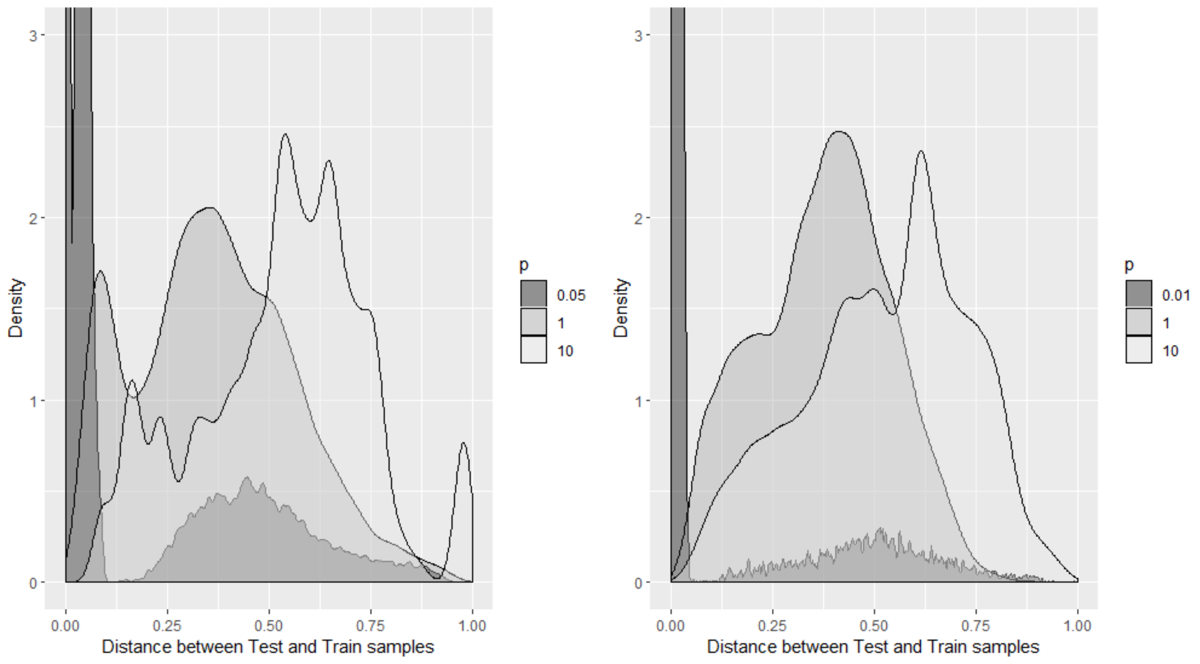

Figure (8) Density plot of the all distances between train and test points for different values of p.

Summarizing, this section showed that there exist a instance selection process for $p \leq 1$ that "remove" points that have variability in more than one dimension with respect to the local axis. Then, based on those reduced set of points the algorithm execute the standard selection rule. As we observed on diverse data sets, this selection process increase the accuracy of the algorithm substantially.

4.3.5 Neighborhood Components Analysis

The fact of recalculating the distance matrix for each value of p, a-priori seem to be a limitation of the model. Therefore, in this section we will compare the methodology optimizing p via hyperparameter search with the metric learning algorithm Neighborhood Component Analysis (NCA) [5].

The objective of NCA is to find the optimal linear transformation $\vec{x}' = L\vec{x}$ such that it maximizes a stochastic variant of the leave-one-out KNN score on the training set. They use a stochastic approach in order to avoid the non differenciable nature of KNN objective functions due to the nearest neighbor discontinuity. They fix the value of p = 2 (Euclidean) and search for optimal matrix L.

In table 7, we highlight the data sets where we found that the norm dependency is strong and we compare the performance of HSW combined with NCA vs empirical search for p.

NCA vs Hyperparameter Search				
Dataset	HSW_p	HSW_L	KNN_p	KNN_L
Ionosphere	94.23 ± 1.05	**95.38**±0.76	90.89 ± 2.45	90.34 ± 1.51
Sonar	82.61 ± 1.61	**88.91**±1.89	81.54 ± 1.42	86.08 ± 1.76
Wine	96.61 ± 1.68	**98.89**±0.53	96.44 ± 0.95	96.77 ± 1.04
Balance Scale	90.01 ± 1.05	**96.05**±0.67	88.67 ± 0.88	95.01 ± 0.90

Table (7) NCA applied to kNN and HSW compared to the same models optimizing p via cross validation. The subscript L indicate NCA and p the typical CV approach. NCA has been applied to the train set to learn the matrix L and then we projected the test samples on that space.

Note that NCA boost substantially the performance of HSW. In addition, HSW is always a few accuracy points above KNN and that in Ionosphere and Wine the increase is significant.

The fact that HSW is closely related to KNN, means that HSW should be able to benefit from metric learning algorithms that aim to optimize KNN. We showed that it is the fact for NCA but it could also work on LMNN (Large Margin Nearest Neighbor) [14].

The main limitation of NCA and LMNN is that the loss functions involve cross product between the matrix L and the points X. This operation scales at most as $O(NP^2)$ which implies that high dimensional data sets are prohibited. We say at most because we use R function *tcrossprod()* which speed up the calculation with the GPU but still the computation is unfeasible. Indeed, NCA will calculate the cross-product for each iteration of the algorithm which enlarge even more the computation time. Thus, in high dimensions, it is couple of orders of magnitudes faster to calculate the distance matrix for each value of p than using metric learning algorithms that use the linear transformation L.

More precisely, let C_L denote the time complexity of NCA, C_L the time complexity of hyperparameter search, N_{iter} the number NCA iterations until convergence and N_{grid} the of different hyperparameters that we want to try. Then the complexity ratio between the two methods can be defined as follows:

$$\frac{C_L}{C_p} \approx \frac{\mathcal{O}(N_{iter} \cdot N \cdot P^2)}{\mathcal{O}(N_{grid} \cdot P \cdot N^2)} = \mathcal{O}\left(\frac{N_{iter} \cdot P}{N_{grid} \cdot N}\right) \tag{13}$$

In general the number of iterations of the NCA is bigger than the number of different values of p that we want to test. However, even if they would be of the same order of magnitude in high dimensional data $P >> N$. For instance in the GEMLeR repository, P is of the order 10^4 where N is of order 10^2. Thus, using p as a hyper-parameter is 100 times faster than linear transformation metric learning. On the other hand, there is also a memory storage problem. In the case of NCA, we need to store a matrix of dimension P x P whereas in the other case we need to store N_{grid} matrices of dimension N x N. Going back to our data sets, NCA will need to store a matrix of 8 bytes x 10000 x 10000 = 800 Mb whereas the hyper-parameter search need to store N_{grid} matrices of 8 x 100 x 100 = 80 Kb. Thus, unless we want to try 10^4 different values of p, the second method is more efficient.

4.3.6 Asymptotic behavior

This section complement the theoretical results from section 4.1 with some simulations.

First of all, in figure 9 we show the cross validation error for different values of k and λ on the Sonar dataset. The discontinuous lines indicate the kNN error rate for the corresponding values of k. The values of k are odd numbers in order to avoid inconsistencies due to ties in the nearest neighbor majority vote. In this particular dataset, since there are no repeated points, note that the limit exist for all exposed values of k.

In order to illustrate the discontinuity of the classifier, we use the synthetics dataset explained in section 4.3.1. On this dataset we analyze the percentage of test points that fall outside the spheres for different values of lambda. More precisely, the faction of test points that have S(x) = 0 (eq. 5).Recall that in order to have a well defined limit, there should exist a value of $\lambda > 0$ such that all points end up in the KNS region. The objective is to show that when we have no repeated points (test points match with centroids) the limit exist and that the repeated points create the discontinuity. In figure 10 we can clearly identify that no matter how small we make lambda, the percentage does not converge to 100 % unless the Oversampling Factor (OF) is exactly equal to 0.

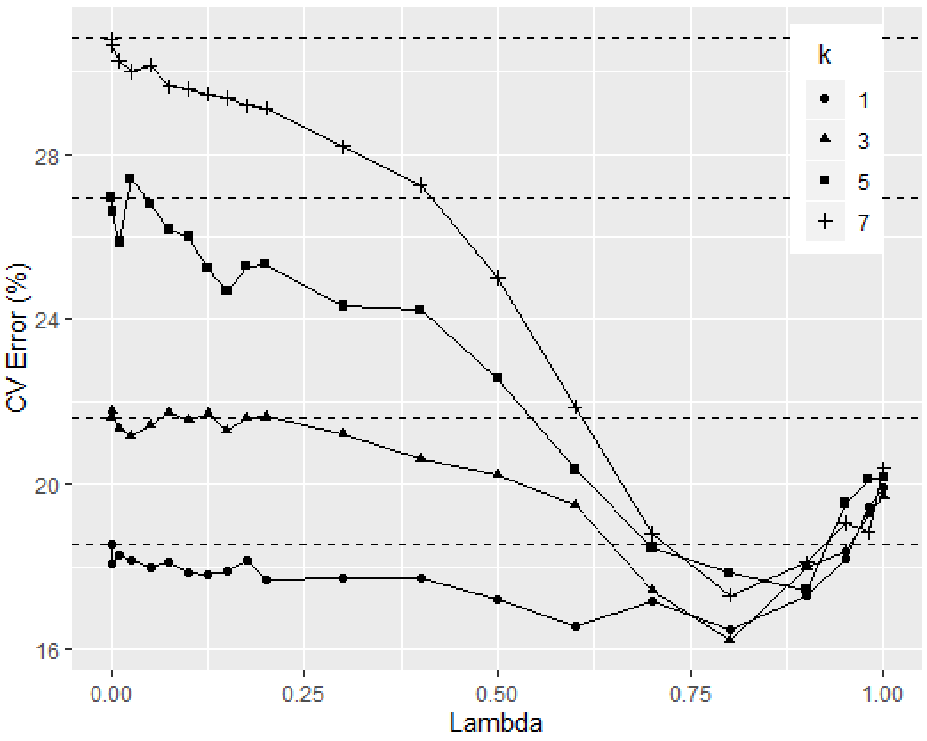

Figure (9) CV error for different values of the hyperparameter lambda and k. The discontinuous lines show the CV error corresponding to the classic KNN algorithm. Data correspond to sonar data-set whereas 10-fold CV and 20 bootstrap samples have been used for the calculations.

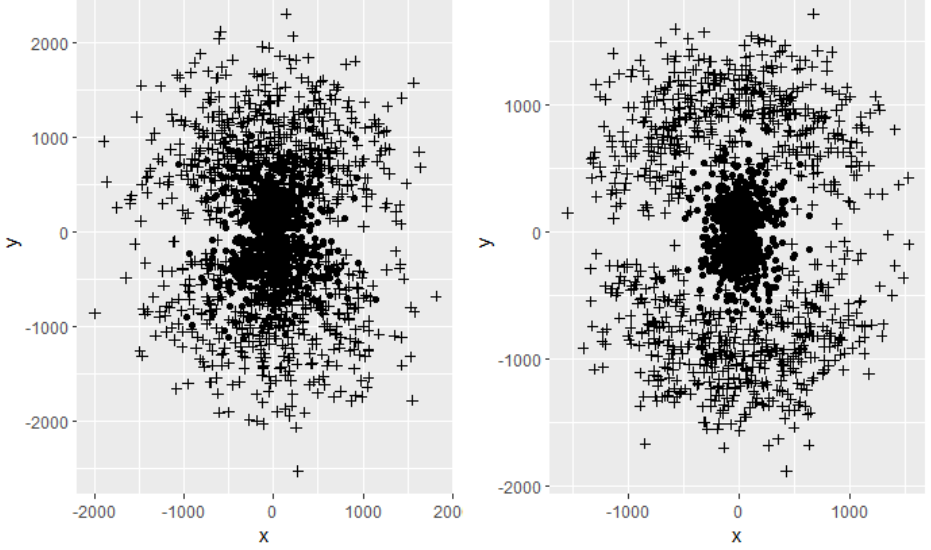

Figure (10) Synthetic dataset with gaussian distribution over the radius of the points. $\sigma_R = 4$ (left) and $\sigma_R = 6$ (rigth). Note that we took the absolute value in order to avoid negative radius.

Finally, we analyze the CV error on this same dataset for different values of the oversampling factor. In figure 12, the discontinuous lines indicate the CV error for KNN and the shapes the CV error for HSW. Again, note that the limit only exist across all values of k when we have no repetition (OF = 0). In the cases when there is heavy repetition of points, both algorithms strongly diverge. To our benefit, HSW shows much stronger performance in the limit of small lambda due to the locality of k. The idea is that HSW is able to detect that two points are exactly equal and therefore it uses locally k = 1 whereas the rest of the points that have more uncertainty it uses the general value of k. This can be one another rule of thumb where HSW outperform KNN: when there is heavy repetition of points in the dataset (many integer features for instance), we can take advantage of the locality of k.

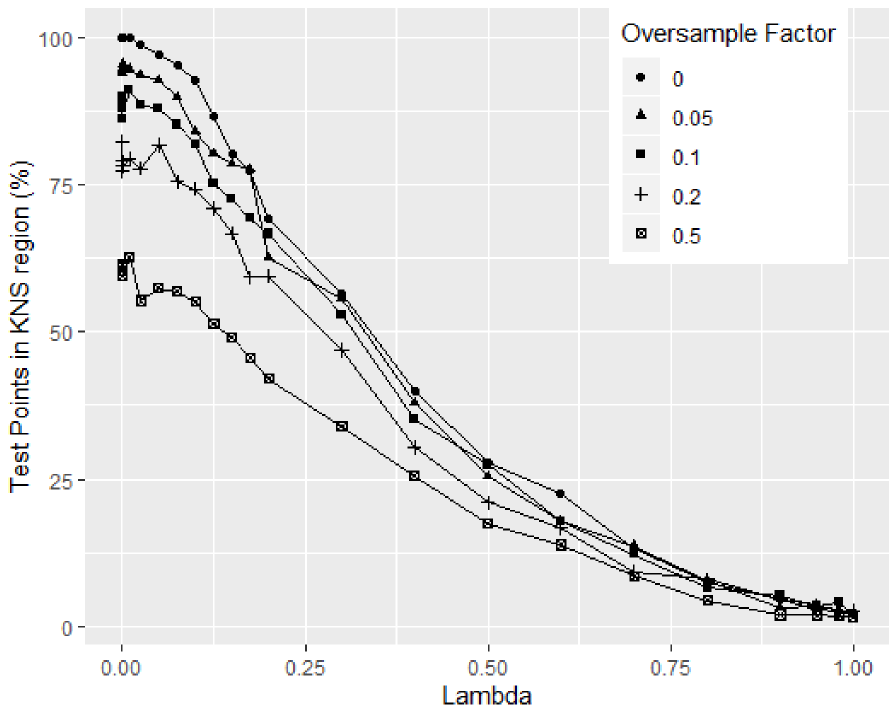

Figure (11) Percentage of test cases that fall into the K-Nearest Sphere (KNS) case for different values of the oversample factor.

Another detail to note in figure 12 is that for k = 1, the limit convergence is satisfied even if there is repetition of points. However, this is only true because the repeated points appear exactly twice. Remember that the value of the local k is determined by the number of times a centroid is repeated. In our case, since the centroids have at most 1 duplicate the local value of k is 1. Since this value coincide with the global k, HSW converge to KNN in the limit of small lambda. Bear in mind that as soon as some point is repeated more than once, the local value of k will no longer coincide and therefore the discontinuity will reappear.

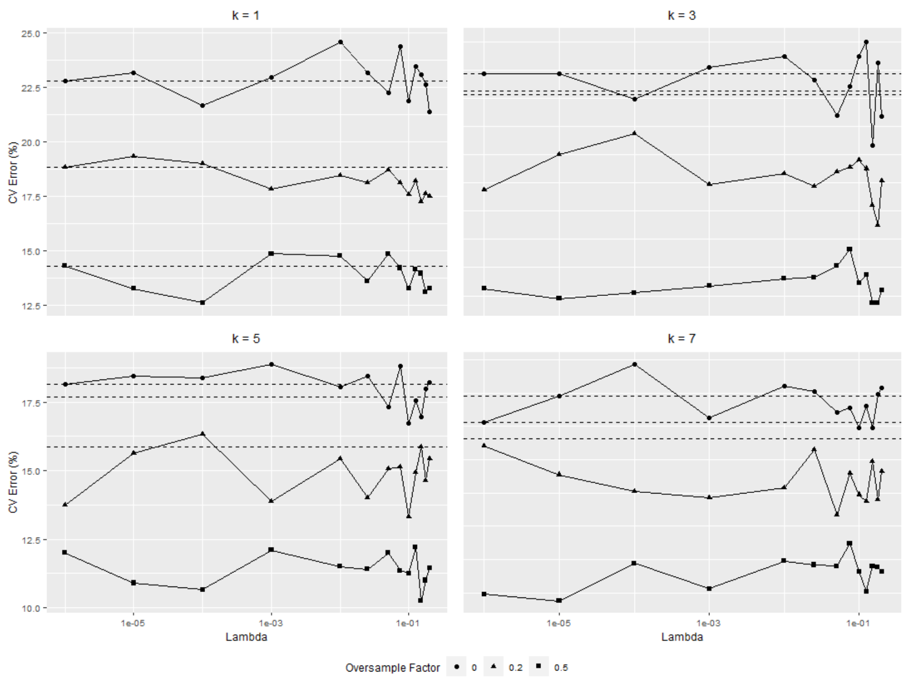

Figure (12) Percentage of test cases that fall into the K-Nearest Sphere (KNS) case for different values of the oversample factor.

5 Related Work

HSW's core idea is to build a classifier that expand spheres around points whose radius depend on the nearest miss. From this point of view, the closest related work that has been found in literature is Large Margin Nearest Neighbors [9]. LMNN approach is to train a mahalanobis metric with the goal that the k-nearest neighbors always belong to the same class while examples from different classes are separated by a large margin. In consequence, samples that belong to the same class will be "compressed" into a hyper sphere and the others will be pushed outside. However, instead of having a single radius, LMNN have a margin that intend to separate the nearest misses from the inner class.

The difference with this work is that HSW does not learn any metric. The union and intersection of hyperspheres shape the boundaries of the classifier with the standard minkovsky metric. Both models attack similar concepts but LMNN from the metric perspective and HSW from the classifier rule perspective. Indeed, it would be interesting to combine HSW and LMNN since the leaned metric in LMNN enhance spherical shaped decision boundaries. Indeed we showed that coupling HSW with NCA [5] add significant improvements over kNN.

Changing the point of view, HSW can be approximated as mixture between two different configurations of nearest neighbors classifiers. A partition of the test set will be predicted with the KNS approach and the other with what we defined in this work as Majority Vote. Even more, the KNS approach can be viewed as a classical kNN rule where the distance from test to train points is replaced by $d(x, c_i) - r_i$. On the other hand, Majority Vote rule can be viewed as a local k nearest neighbor model replacing the distances by $\min(d(x, c_i) - r_i, 0))$. Specifically, the nearest neighbors are defined by the number of spheres that contain the test point.

Therefore, we can decompose HSW in three related work areas:

- **Mixture of kNN models**. The is plenty of literature that combine kNN with other models. In order to speed up computation time [15] used kNN combined with K-means clustering algorithm. In addition, [16] proposed a hybrid version of KNN with SVM for visual category recognition. Also kNN has been combined with Genetic Algorithms, Ant Colony Optimization or Swarm Optimization among many others [3]. Nonetheless, to the best of our knowledge there is no related work that combine different configurations of kNN in a single model.
- **Similarity measure selection**. It is well known that the similarity measure has strong impact on the nearest neighbor classifier. Indeed, [17] studies the performance of NN rule with over 50 different similarity measures on 30 data sets from the UCI Machine Learning Repository. They show that the Hassanat Distance gives the best average result on those datasets. In any case, we haven't found any related work that use a distance metric that involve the nearest misses of the train points.
- **Local adaptive k value**. For instance [18] calculate the best local value of k for each instance in the train set with cross validation. Then, for each test point they use the value of k corresponding to the average 3 nearest neighbors. In order to add a smoothing factor and break ties they also take into consideration the impact of the local k to the global accuracy. Another example is Adaptive kNN algorithm [19]. In their work the optimal value of k is defined as then number of the fewest nearest neighbors a training example has to identify to get its correct class label. In comparison with HSW, the local value of k arise only as a consequence from the distance metric and how the radius depend with the nearest misses.

6 Conclusions

This work showed that HSW is able to compete with other well know algorithms like SVM, kNN, XGboost or Logistic Regression. In general terms, this only applies to Multi-Class and Complete HSW whereas One-Class HSW is not able to reach a solid benchmark in classification. However, on low dimensional sets, we cannot guarantee that the complete version will always be better than multi-class. A randomized grid search should be enough in those cases. Regarding structure affinity for HSW, it seems that when at least one of the classes has a high spherical symmetry, the algorithm show strong performance. In addition, it would be worth to try HSW when we have a high number o repeated points such that we can benefit from the locality of k.

The structure of HSW is closely related to kNN. Indeed, the work shows that complete HSW converge to kNN in the limit $\lambda \to 0$ when there are no repeated points in the dataset. In contrast, the presence of repeated points imply that HSW converge to a local kNN rule where the value of k_i is exactly equal to the number of repeated points such that $x_i = c_i$. In any case, since the value $\lambda = 0$ always recover the classical kNN approach, we can guarantee than HSW will always perform better or equal than kNN. The magnitude of the improvement will depend on how well this linear second order interaction suits the data.

Due to the time complexity of the algorithm, we investigated the performance on high dimensional data. In almost all studied datasets, HSW perform significantly better than kNN with p = 2. Perhaps most importantly, in some cases HSW is also significantly better than kNN with optimal p which indicate that the second order interaction add value to kNN. On the other hand, we found little differences with Logistic Regression and Linear SVM on the studied dataset.

In addition, HSW required very low values of p in all the studied high dimensional datasets. Those values of p create a instance selection process that exclude points with variability in more than one direction from the nearest neighbor selection. It is yet unclear why exactly this process boost the accuracy of HSW and kNN and if this is purely an effect of dimensionality.

Finally, the work also shows that metric learning algorithms like neighborhood component analysis are able to boost substantially the performance of HSW. This technique show much better results than searching for p in the hyper parameter space via cross validation. However, those metric learning processes are unfeasible in high dimensional data. Indeed, in those cases it is much more efficient to recalculate the matrix distance for each value of p.

7 Future Work

Currently, the function that determine how the radius of the spheres depend with the nearest miss is defined as a linear relationship $r_i = \lambda U_i$. Other functions of interest could be involving the furthest hit V_i such that we radius is defined as $r_i = \lambda(U_i + V_i)/2$. We understand the furthest hit as the most distant positive sample before the nearest miss. This function create a margin that should be is less biased towards the class of the sphere. Indeed, those functions can be probabilistic in the sense that we assume a distribution over values of r_i. For instance a valid approach could be to assume that r_i follow a normal distribution with mean value $\mu = (U_i + V_i)/2$ and variance $\sigma^2 = \lambda$ where now the hyper-parameter define the uncertainty of the margin.

On the other hand, it could be interesting to studying if Kth order interactions (K > 2) are able to add value to the model. For instance a 3rd order interaction could be that the selection rule depend not only on the distance and nearest miss but also on the nearest hit of the nearest miss. In that case KNS distance criterion would be $d(x, c_i) - U_i + W_j$ where W_j represent the nearest hit of U_i. In that case, the rule would take into account if the nearest miss if far away from other samples of the same class or not. This idea could be expanded to any K such that for a Kth order interaction the radius array R would become a (K-1,N) matrix. First row represent the nearest miss of each data point, second row the nearest hit of the point in row 1, third row the nearest miss of the point calculated in row 2, and so on.

Another topic of interest could be study the kernelized HSW rule and if it is a potential alternative to the grid search of p. In the same direction, a metric learning algorithm that optimize p instead of the matrix L could be a valuable alternative in terms of scalability in high dimensional data.

To conclude, remark that most works dedicated to improve kNN scalability [6][19][20][21] could be also applied to HSW, at least to the predict procedure and the distance matrix computation. On the other hand, the train procedure of calculating the nearest miss can be easily parallelized since the calculations for each sample are independent.

References

[1] E. Fix, *Discriminatory analysis: nonparametric discrimination, consistency properties.* USAF School of Aviation Medicine, 1951.

[2] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[3] N. Bhatia *et al.*, "Survey of nearest neighbor techniques," *arXiv preprint arXiv:1007.0085*, 2010.

[4] R. J. Samworth *et al.*, "Optimal weighted nearest neighbour classifiers," *The Annals of Statistics*, vol. 40, no. 5, pp. 2733–2763, 2012.

[5] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov, "Neighbourhood components analysis," in *Advances in neural information processing systems*, pp. 513–520, 2005.

[6] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematical Software (TOMS)*, vol. 3, no. 3, pp. 209–226, 1977.

[7] *Topology.* Upper Saddle River: Prentice Hall, 2000.

[8] D. M. Tax and R. P. Duin, "Support vector data description," *Machine learning*, vol. 54, no. 1, pp. 45–66, 2004.

[9] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification.," *Journal of Machine Learning Research*, vol. 10, no. 2, 2009.

[10] Y. Zhu, Z. Wang, and D. Gao, "Gravitational fixed radius nearest neighbor for imbalanced problem," *Knowledge-Based Systems*, vol. 90, pp. 224–238, 2015.

[11] E. W. Weisstein, "Vector norm."

[12] K. Kira, L. A. Rendell, *et al.*, "The feature selection problem: Traditional methods and a new algorithm," in *Aaai*, vol. 2, pp. 129–134, 1992.

[13] I. J. Maddox, *Elements of functional analysis.* CUP Archive, 1988.

[14] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification.," *Journal of Machine Learning Research*, vol. 10, no. 2, 2009.

[15] P. W. Buana, S. Jannet, I. Putra, *et al.*, "Combination of k-nearest neighbor and k-means based on term re-weighting for classify indonesian news," *International Journal of Computer Applications*, vol. 50, no. 11, pp. 37–42, 2012.

[16] H. Zhang, A. C. Berg, M. Maire, and J. Malik, "Svm-knn: Discriminative nearest neighbor classification for visual category recognition," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 2126–2136, IEEE, 2006.

[17] V. Prasatha, H. A. A. Alfeilate, A. Hassanate, O. Lasassmehe, A. S. Tarawnehf, M. B. Alhasanatg, and H. S. E. Salmane, "Effects of distance measure choice on knn classifier performance-a review," *arXiv preprint arXiv:1708.04321*, 2017.

[18] N. García-Pedrajas, J. A. Romero del Castillo, and G. Cerruela-García, "A proposal for local k values for k -nearest neighbor rule," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 2, pp. 470–475, 2017.

[19] S. Sun and R. Huang, "An adaptive k-nearest neighbor algorithm," in *2010 seventh international conference on fuzzy systems and knowledge discovery*, vol. 1, pp. 91–94, IEEE, 2010.

[20] V. Garcia, E. Debreuve, and M. Barlaud, "Fast k nearest neighbor search using gpu," in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1–6, IEEE, 2008.

[21] Y. Wu, K. Ianakiev, and V. Govindaraju, "Improved k-nearest neighbor classification," *Pattern recognition*, vol. 35, no. 10, pp. 2311–2318, 2002.