CS-E4600- Algorithmic methods of datamining
Home Assignment 2 - Ville Virkkala 63325V

Problem 1: Hausdorff distance

Answer Q1.1.
In the Hausdorff distance the minimum distance for each point in $x \in A$ from the set B is calculated. After that a maximum of those distances is selected as the Hausdorff distance. To make the hausdorff distance proper metric, the symmetry is induced by calculating the same maximum distance when $x \in B$ and the distance is calculated with respect to A instead of B. Finally maximum of the obtained distances is taken as the proper Hausdorff metric. The hausdorff distance $d_H(A, B)$ descripes the maximum distance between any point of one set to the other set and is more general than, for example, the minimum distance between sets that applies to only on point in set.
Answer Q1.2.
Here the L_∞ of $f(A)$ is directly the definition of non-symmetric hausdorff distance where $d(x_i, A)$ is the minimum distance from point x_i to A and finally the L_∞ selects the largest of those distances which is the non-symmetric Hausdorff distance described above.
Answer Q1.2.

(a) $d_H(A, B) \geq 0$
Because the metric $d : X \times X$ is proper metric for which $d(x, y) \geq 0$ for all $x, y \in X$ then clearly $d_H(A, B) \geq 0$ for all $A, B \subseteq X$.

(b) $d_H(A, B) = 0$ if and only if $A = B$
If $A = B$ then $d_H(A, B) = 0$, because $D(x, B) = D(x, A) = 0, \forall x \in A$. If $d_H(A, B) = 0$ then every element of A is at zero distance from elements of B and thus $A \subset B$ and the same is true for B and thus $A = B$.

(c) $d_H(A, B) = d_H(B, A)$
The condition $d_H(A, B) = d_H(B, A)$ is valid by definition for proper symmetrized Hausdorff metric

(d) $d_H(A, C) \leq d_H(A, B) + d_H(B, C)$
For single point $a \in A$ it holds $d_H(a, B) \leq \operatorname*{argmin}_{b \in B} d(a, b)$. Because d was a proper metric it holds $d(a, b) \leq d(a, c) + d(c, b), \forall c$ in C and thus

$$d_H(a, B) \quad \leq \quad \operatorname*{argmin}_{b \in B}(d(a, c) + d(c, b)).$$

Because $d(a, c)$ is independent of B above equation can be written as

$$\begin{aligned} d_H(a, B) &\leq d(a, c) + \operatorname*{argmin}_{b \in B} d(c, b), \qquad \operatorname*{argmin}_{b \in B} d(c, b) \leq d_H(c, B) \\ &\leq d(a, c) + d_H(c, B), \qquad d(a, c) \leq \operatorname*{argmax}_{c \in C} d(a, c) = d_H(a, C) \text{ and } d_H(c, B) \leq d_H(C, B) \\ &\leq d_H(a, C) + d_H(C, B). \end{aligned} \tag{1}$$

Now maximizing both sides in eq. (1) with respect to a yields

$$\begin{aligned} \operatorname*{argmax}_{a \in A} d_H(a, B) &\leq \operatorname*{argmax}_{a \in A}(d_H(a, C) + d_H(C, B)) \\ d_H(A, B) &\leq d_H(A, C) + d_H(C, B). \end{aligned} \tag{2}$$

Repeating above derivation for $d_H(b, A)$ completes the proof.

Problem 2: Locality sensitive hashing

To show that $f_{\vec{r}}$ is a locality sensitive hashing of $s(x, y)$ we must show that $P(f_{\vec{r}}(\vec{r}\cdot\vec{x}) = f_{\vec{r}}(\vec{r}\cdot\vec{y})) = s(x, y)$. Lets start with the case $d = 2$. Lets consider vector $\vec{x}$ that is oriented along the y-axis. Then, as visualized in figure 1, the signum is positive for vectors $\vec{r}$ that are on the same side as $\vec{x}$ with respect to line perpendicular to $\vec{x}$ and negative on the opposite side. Now because the dot product between two vectors is rotation invariant, the above is true for all vectors $\vec{x}$, i.e., $\mathrm{sign}(\vec{r}\cdot\vec{x})$ is negative for vectors $\vec{r}$ that are on the opposite side of the perpendicular line to $\vec{x}$ than $\vec{x}$ and positive on the same side. Now if we take a vector $\vec{y}$ and rotate it with angle α with respect to $\vec{x}$ we see that the $\mathrm{sign}(\vec{r}\cdot\vec{x}) \neq \mathrm{sign}(\vec{r}\cdot\vec{y})$ when $\vec{r}$ lies between the lines perpendicular to $\vec{x}$ and $\vec{y}$, corresponding to angle $\theta \in [0, \alpha]$, and $\mathrm{sign}(\vec{r}\cdot\vec{x}) = \mathrm{sign}(\vec{r}\cdot\vec{y})$ when $\theta \in [\alpha, \pi]$ as shown in figure 1. Thus the probability that $f_{\vec{r}}(\vec{r}\cdot\vec{x}) = f_{\vec{r}}(\vec{r}\cdot\vec{y})$ is $\frac{1}{\pi}(\pi - \alpha) = 1 - \frac{\alpha}{\pi} = s(x, y)$. For higher dimensions d the vectors x and y span a two dimensional subspace $S \subset \mathbb{R}^d$ and the angle between the vectors $\vec{x}$ and $\vec{y}$ is defined in this plane. Now the vector $\vec{r}$ can be composed to vectors $\vec{r}_{\|}$ parallel to S and $\vec{r}_{\perp}$ perpedicular to S. Now we got for the dot product between $f_{\vec{r}}(\vec{r}\cdot\vec{x})$ in $\mathbb{R}^d$

$$\begin{aligned} f_{\vec{r}}(\vec{r}\cdot\vec{x}) &= f_{\vec{r}}((\vec{r_{\|}} + \vec{r_{\perp}})\cdot\vec{x}) \\ &= f_{\vec{r}}(\vec{r_{\|}}\cdot\vec{x}). \end{aligned} \tag{3}$$

Now because the angle between $\vec{x}$ and $\vec{y}$ is defined in the plane S and the vector $\vec{r_{\|}}$ lies on the same plane the problem reduces to the 2d-dimensional case described above and thus the equality $P(f_{\vec{r}}(\vec{r}\cdot\vec{x}) = f_{\vec{r}}(\vec{r}\cdot\vec{y})) = s(x, y)$ hold also for $\mathbb{R}^d$.

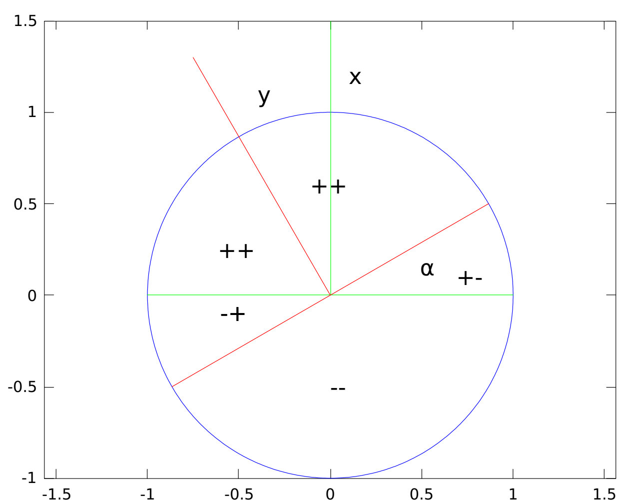

Figure 1: Visualization of the signum $f_{\vec{r}}(\vec{r}\cdot\vec{x})$. Because the dot product is rotation invariant the signums remains same relative to x and y under every rotation α of y

Problem 3: Sliding window

Assuming that X contains do duplicates (drawn randomly from uniform distribution). Then at the first step the maximum is updated for sure. For the next step we have two items and the probability that the new item is the larger one is $\frac{1}{2}$. Continuing this way for $i = 1, 2, 3, 4 \ldots, n$ we get for the number of updates

$$N = \sum_{i=1}^{m} \frac{1}{i}. \tag{4}$$

For large m the equation 4 can be approximated by $\log(m)$. For the latter part of the problem I do not know the proof.

Problem 4: Frequence of item i in sequence

Answer Q4.1.
A simple approach is to create a vector a of length n which keeps count of times every $i \in 1, 2, \ldots n$ has occured in X, *i.e.*, $a(i) = m_i$ the number of times i occurred in X. Now we can compute m_i for all $i \in 1, 2, \ldots n$ by looping over the array X and increasing the counter $a(x_j) = a(x_j) + 1$ at every step j. The maximum times i can occur in X is m. To store an integer m requires $\log(m) + 1$ bits. Thus required memory for the array a is $\mathcal{O}(n\log(m))$.

Answer Q4.2.
Let $s[x_i] = s_i$ and frequence of item i be f_i. We got for the expectation of $E[c \cdot s[x_i]]$

$$\begin{aligned} E[c \cdot s[x_i]] &= E[(f_1 s_1 + f_2 s_2 + \ldots + f_i s_i + \ldots + f_n s_n) s_i] \\ &= E[\sum_{j \neq i} f_j s_j s_i] + E[f_i s_i^2] \\ &= \sum_{j \neq i} f_j E[s_j] E[s_i] + f_i E[s_i^2] \\ &= f_i. \end{aligned} \tag{5}$$

In above we use the fact $E[s_j] = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot (-1) = 0$ and $E[s_i^2] = \frac{1}{2} \cdot 1^2 + \frac{1}{2} \cdot (-1)^2 = 1$.

Problem 5: Independent random variables

Lets first show $E(X1 + X2) = 2\mu$.

$$\begin{aligned} E(X_1 + X_2) &= \sum_{x_1} \sum_{x_2} (x_1 + x_2) P_{X_1 X_2}(x1, x2) \\ &= \sum_{x_1} x_1 \sum_{x_2} P_{X_1 X_2}(x1, x2) + \sum_{x_1} \sum_{x_2} x_2 P_{X_1 X_2}(x1, x2) \qquad \text{order of summation can be changed} \\ &= \sum_{x_1} x_1 \sum_{x_2} P_{X_1 X_2}(x1, x2) + \sum_{x_2} x_2 \sum_{x_1} P_{X_1 X_2}(x1, x2) \\ &= \sum_{x_1} x_1 P_{X_1}(x1) + \sum_{x_2} x_2 P_{X_2}(x2) \\ &= 2\mu. \end{aligned} \tag{6}$$

Now using induction we can show that $E(X_1 + X_2 + \ldots X_k) = k\mu$. Thus

$$\begin{aligned} E(\frac{1}{k}(X_1 + X_2 + \ldots + X_k)) &= \frac{1}{k}E(X_1 + X_2 + \ldots + X_k) \\ &= \frac{1}{k}k\mu \\ &= \mu. \end{aligned} \tag{7}$$

If the random variables $X_1, X_2, \ldots, X_n$ are continuous we simply replace the summation by integral. For the variance $Var(X_1 + X_2)$ we get in a similar way

$$\begin{aligned} Var(X_1 + X_2) &= E((X_1 + X_2)^2) - E(X_1 + X_2)^2 \\ &= \sum_{x_1}\sum_{x_2}(x_1 + x_2)^2 P_{X_1X_2}(x1, x2) - E(X_1)^2 - E(X_2)^2 - 2E(X_1)E(X_2) \\ &= \sum_{x_1} x_1^2 \sum_{x_2} P_{X_1X_2}(x1, x2) + \sum_{x_2} x_2^2 \sum_{x_1} P_{X_1X_2}(x1, x2) + 2\sum_{x_1}\sum_{x_2} x_1x_2P_{X_1X_2}(x1, x2) \\ &\quad -E(X_1)^2 - E(X_2)^2 - 2E(X_1)E(X_2) \\ &= E(X_1^2) - E(X_1)^2 + E(X_2^2) - E(X_2)^2 + 2E(X_1X_2) - 2E(X_1)E(X_2) \\ &= Var(X_1) + Var(X_2), \end{aligned} \tag{8}$$

and thus $Var(X_1 + X_2 + \ldots + X_k) = k\sigma^2$. In above the property $E(X_1X_2) = E(X_1)E(X_2)$ for independent variables was used. Now for $Var(Y)$ we get

$$\begin{aligned} Var(Y) &= Var(\frac{1}{k}\sum_{i=1}^{k} X_i) \\ &= \frac{1}{k^2}Var(\sum_{i=1}^{k} X_i) \\ &= \frac{1}{k^2}k\sigma^2 \\ &= \frac{1}{k}\sigma^2. \end{aligned} \tag{9}$$

Problem 6: Max-match distance

Solutions presented in this section are heavily based on the following sources `https://en.wikipedia.org/wiki/Hausdorff_distance` and `http://cgm.cs.mcgill.ca/~godfried/teaching/cg-projects/98/normand/main.html`

Answer Q4.1.
One possible example of an object X would be a graph. In this case U would be the set of all possible vertices and d would describe the distance between two vertices.

Answer Q4.2.
The max-match distance $d(A, B)$ described above is the at most distance of any vertex in A from vertices of B. Another possibility would be calculate the minimum distance between points of

A and B. However this distance is sensitive to the orientation of graphs A and B and thus the max-match distance is more descriptive.

Answer Q4.3.
The max-match distance is not a proper metric because requirement $d(A, B) = d(B, A)$ is not fulfilled. However an induced metric $d'(A, B) = max\{d(A, B), d(B, A)\}$ is a proper metric as stated in `https://en.wikipedia.org/wiki/Hausdorff_distance`.

Answer Q4.4.
The max-match algorithm could be implemented using the algorithm 1 (see next page) which running time scales as $\mathcal{O}(n^2)$

Algorithm 1 Max-match algorithm

Input: graphs $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, , \ldots, y_n\}$, Euclidean-distance d
Output: Max-match distance $dmaxmatch$ between graphs X and Y

```
maxmatch ← 0
for i = 1, ..., n do
    dmin ← d(x_i, y_1)
    for j = 2, ..., n do
        dtmp ← d(x_i, y_j)
        if dtmp < dmin then
            dmin ← dtmp
        end if
    end for
    if dmin > dmaxmatch then
        dmaxmatch ← dmin
    end if
end for
return dmaxmatch
```

Answer Q4.5.
Instead of calculating the exact $D(X, Y)$ one could select a particular subset of points $X_S \in X$ and calculate the distance $D(X_s, Y)$ instead of $D(X, Y)$. For this distance it is guaranteed that $D(X_s, Y) <= D(X, Y)$. One possible subset could be the points of X that are located on the minimum bounding box of X.

Answer Q4.6.
An algorithm that calculates the lower bound distance using the above description can be obtained by calculating points located on the minimum bounding box of X, $X_S = mbb(X)$, and giving the resulting set X_s as input to algorithm 1 instead of X. This algorithm would scale as $\mathcal{O}(2^d n)$ where d is the dimension of data-points in X and Y and n is the number of points in Y.