

Υπολογιστική Γεωμετρία  
Δεύτερη Εργασία

Σιώρος Βασίλειος - 1115201500144  
Ανδρινοπούλου Χριστίνα - 1115201500006

Απρίλιος 2020

# 1. Implement from scratch the k-NN algorithm in python programming language. Present your method and how you worked, conclude by discussing the disadvantages of the k- NN algorithm, if any.

## Thought Process and Theory

Πολύ συχνά, σε διάφορους τομείς της επιστήμης, αλλά και της καθημερινότητας, χρειάζεται να προβλέψουμε συμπεριφορές βασιζόμενοι σε ήδη γνωστά στοιχεία. Ένα τέτοιο πολύ απλό παράδειγμα μπορεί να είναι η πρόβλεψη του βαθμού των φοιτητών στο μάθημα της Υπολογιστικής Γεωμετρίας με βάση τους βαθμούς των φοιτητών των προηγούμενων χρόνων. Στο παράδειγμα αυτό έχουμε ένα σύνολο φοιτητών για τους οποίους γνωρίζουμε τη βαθμολογία τους στο μάθημα (φοιτητές προηγούμενων χρόνων) και ένα σύνολο φοιτητών (φοιτητές τρέχοντος εξαμήνου) για τους οποίους θέλουμε να προβλέψουμε τον βαθμό που θα πάρουν. Η μηχανική μαθησης και η εξόρυξη δεδομένων διαχειρίζεται τέτοια προβλήματα, ανεξάρτητα από τη φύση τους, μέσα από συγκεκριμένους αλγορίθμους. Ένας τέτοιος αλγόριθμος είναι ο αλγόριθμος των  $k$ -πλησιέστερων γειτόνων.

Ο αλγόριθμος των  $k$ -πλησιέστερων γειτόνων (**k**-Nearest Neighbor - **k**-NN) είναι ένα κλασσικός αλγόριθμος μηχανικής μάθησης. Η βασική ιδέα του αλγορίθμου αυτού είναι η πρόβλεψη της κατηγορίας ενός αντικειμένου με βάση τις κατηγορίες αντικειμένων που είναι ήδη γνωστές στον αλγόριθμο. Πιο συγκεκριμένα, ο **k**-NN έχει να διαχειριστεί δύο διαφορετικά σύνολα αντικειμένων, εκείνα τα αντικείμενα για τα οποία γνωρίζει την κατηγορία (**label**) στην οποία ανήκουν και εκείνα για τα οποία αγνοεί πλήρως την κατηγορία στην οποία εντάσσονται και επιχειρεί να την προβλέψει.

Κάθε αντικείμενο περιγράφεται από ένα σύνολο χαρακτηριστικών. Χαρακτηριστικό καλείται μία ιδιότητα ή ένα γνώρισμα ενός αντικειμένου. Η ιδιότητα (ή το γνώρισμα) μπορεί να παρουσιάζει διαφοροποιήσεις από αντικείμενο σε αντικείμενο ή από χρονική στιγμή σε χρονική στιγμή. Ενδεικτικά, για την περιγραφή του αντικειμένου "άνθρωπος" σε ένα πρόβλημα βιοϊατρικού χαρακτήρα μερικά από τα χαρακτηριστικά που θα το περιέγραφαν μπορούν να είναι: το φύλο, η ηλικία, το βάρος, οι ασθένειες από τις οποίες πάσχει, οι ασθένειες από τις οποίες πάσχουν/έπασχαν οι συγγενείς του, το είδος διατροφής κ.α.. Το πλήθος των χαρακτηριστικών καθορίζει τον χώρο στον οποίον ζουν τα αντικείμενα. Αν για παράδειγμα τα αντικείμενα περιγράφονται από τρία χαρακτηριστικά, τότε τα αντικείμενα εντόπιζονται στον  $\mathbb{R}^3$  και μπορούν να αναπαρασταθούν ως σημεία στον τρισδιάστατο χώρο με  $x$ ,  $y$  και  $z$  συντεταγμένη που να αντιστοιχούν στο πρώτο, δεύτερο και τρίτο χαρακτηριστικό αντίστοιχα. Αν τα χαρακτηριστικά δεν έχουν αριθμητική μορφή μπορούν, ανάλογα με τη φύση του χαρακτηριστικού, να μετατραπούν σε αριθμούς.

Επίσης, κάθε αντικείμενο ανήκει σε μία συγκεκριμένη κατηγορία (**label**). Στο παραπάνω παράδειγμα βιοϊατρικής φύσης η κατηγοριποίηση των ανθρώπων μπορεί να γίνει με βάση την ύπαρξη ή όχι κάποιας συγκεκριμένης ασθένειας σε κάθε άνθρωπο. Συνεπώς, η κατηγορία εδώ είναι "πάσχει/δεν πάσχει" από μία ασθένεια, έστω καρκίνος.

Τα χαρακτηριστικά, όπως αναφέραμε προηγουμένως, χρησιμοποιούνται για να περιγράψουν το εκάστοτε αντικείμενο και είναι συνυφασμένα με την κατηγορία στην οποία εντάσσεται το αντικείμενο με τέτοιον τρόπο ώστε αντικείμενα που μοιάζουν μεταξύ τους να ανήκουν στην ίδια κατηγορία.

Ο αλγόριθμος για να εκκινήσει απαιτεί ένα σύνολο δεδομένων για τα οποία είναι ήδη γνωστές οι κατηγορίες. Το σύνολο αυτό καλείται σύνολο εκπαίδευσης. Έπειτα, με τρόπο που θα περιγραφεί

στη συνέχεια, μπορεί να αποφανθεί για τις κατηγορίες δειγμάτων που έχουν τα ίδια χαρακτηριστικά με εκείνα του συνόλου εκπαίδευσης.

Έστω  $Z$  το αντικείμενο που εξετάζεται ως προς την κατηγορία του. Ο k-NN εντοπίζει όλα τα αντικείμενα του συνόλου εκπαίδευσης που έχουν όμοια ή σχεδόν όμοια χαρακτηριστικά με το  $Z$ . Τα αντικείμενα αυτά καλούνται πλησιέστεροι γείτονες (*nearest neighbors*). Ο αλγόριθμος επιλέγει κάθε φορά να λάβει υπ' όψιν μόνο τους  $k$  πλησιέστερους γείτονες στην πρόβλεψη της κατηγορίας του  $Z$  και αυτό είναι και το χαρακτηριστικό που δικαιολογεί το όνομά του. Ουσιαστικά, υπολογίζει την εγγύτητα του  $Z$  σε σχέση με τα αντικείμενα του συνόλου εκπαίδευσης και επιλέγει μόνο  $k$  το πλήθος από αυτά τα οποία ομοιάζουν περισσότερο με το  $Z$ .

Η εγγύτητα του αλγορίθμου υπολογίζεται σύμφωνα με κάποιο μέτρο ομοιότητας. Ένας κλασικός τρόπος σύγκρισης δύο αντικειμένων είναι ο υπολογισμός της Ευκλείδιας Απόστασής τους. Η Ευκλείδια Απόσταση δύο σημείων, έστω τα  $x$  και  $x'$  σημεία, που βρίσκονται σε χώρο διάστασης  $d$  δίνεται από τον τύπο

$$\rho(x, x') = \|x - x'\| = \sqrt{\sum_{i=1}^d (x_i - x'_i)^2}$$

Το μέτρο της Ευκλείδιας απόστασης γενικεύεται από τον παρακάτω τύπο

$$\rho(x, x') = \left( \sum_{i=1}^d |(x_i - x'_i)|^r \right)^{\frac{1}{r}}$$

ο οποίος είναι το μέτρο της απόστασης Minkowski και για διαφορετικές τιμές του  $r$  δίνει διαφορετικά είδη αποστάσεων:

- για  $r = 1$  : Manhattan απόσταση ( $L_1$ )
- για  $r = 2$  : Ευκλείδια απόσταση ( $L_2$ )
- για  $r = \infty$  : Απόσταση άνω φράγματος ( $L_\infty$ )

Ο k-NN, αφού επιλέξει τους  $k$  πλησιέστερους γείτονες του αντικειμένου  $Z$  που εξετάζει, το κατηγοριοποιεί ανάλογα με τα *labels* των γειτόνων. Αν οι πλησιέστεροι γείτονες ανήκουν όλοι στην ίδια κατηγορία, τότε και το  $Z$  κατηγοριοποιείται σε αυτήν. Ωστόσο, αν οι πλησιέστεροι γείτονες ανήκουν σε παραπάνω από μία κατηγορίες, τότε το  $Z$  λαμβάνει την επικέτα της πλειοψηφίας. Η σχέση που ακολουθεί κωδικοποιεί την τακτική αυτήν του αλγορίθμου.

$$\text{Majority Vote: } y' = \operatorname{argmax} \sum_{(x_i, l_i) \in D_z} I(l = l_i),$$

όπου  $l$  είναι κάποιο *label*,  $l_i$  είναι το *label* ενός γείτονα και  $I$  συνάρτηση που επιστρέφει αληθές αν το όρισμα είναι αληθές και ψευδές σε διαφορετική περίπτωση.

Ο αλγόριθμος των  $k$  πλησιέστερων γειτόνων μπορεί να εκτελεί το καθήκον του, δηλαδή να κατηγοριοποιεί τα αντικείμενα που του δίνονται με βάση όλα ήδη κατηγοριοποιημένα αντικείμενα, χωρίς να απαιτεί επιπλέον χρόνο εκπαίδευσης με βάση το σύνολο εκπαίδευσης. Αύτο τον κάνει

ιδιαίτερα ευέλικτο αλγόριθμος, καθώς μπορεί οποιαδήποτε στιγμή να προσθέσεις και νούρια αντικείμενα προς κατηγοριοποίηση χωρίς να απαιτείται καμία επιπροσθετη ενέργεια. Ωστόσο, ο αλγόριθμος έχει και ορισμένα μειονεκτήματα.

- Η κατηγοριοποίηση κάθε αντικειμένου είναι μία ακριβή διαδικασία, καθώς είναι απαραίτητος ο υπολογισμός της απόστασης του εκάστοτε δείγματος ελέγχου με όλα τα δείγματα του συνόλου εκπαίδευσης. Συνεπώς, αν τα δείγματα ελέγχου ή/και τα δείγματα εκπαίδευσης είναι πολλά σε αριθμό, τότε ο αλγόριθμος πρέπει να δαπανήσει χρόνο στον υπολογισμό των αντίστοιχων μέτρων απόστασης. Ο χρόνος αυξάνει επιπλέον όταν το πλήθος των χαρακτηριστικών των αντικείμενων είναι μεγάλο.
- Ο αλγόριθμος προβλέπει την κατηγορία ενός αντικειμένου σύμφωνα με τις τοπικές πληροφορίες, δηλαδή σύμφωνα με τους γείτονες που βρίσκονται κοντά στο υπό εξέταση αντικείμενο, χωρίς να λαμβάνει υπ' όψιν οτιδήποτε άλλο. Συνεπώς, η επιλογή του  $k$ , δηλαδή η επιλογή του πλήθους των γειτόνων που θα καθορίσουν το label ενός αντικειμένου είναι κρίσιμης σημασίας. Αν επιλεγεί  $k$  που είναι πολύ μικρό σε σύγκριση με τον óγκο των δεδομένων, τότε ο αλγόριθμος θα γίνει επιφρεπής σε θορύβους και θα υπερπροσαρμοστεί στα δεδομένα. Αν επιλεγεί  $k$  το οποίο να είναι πολύ μεγάλο, τότε πιθανότατα η κατηγοριοποίηση που θα γίνει από τον αλγόριθμο να μην είναι ασφαλής, διότι το label κάθε δείγματος ελέγχου θα επηρεάζεται από ένα σύνολο δειγμάτων εκπαίδευσης του οποίου τα περισσότερα δείγματα θα είναι αρκετά απομακρυσμένα από το τρέχον δείγμα ελέγχου και συνεπώς όχι όμοια με αυτό. Αυτό σημαίνει ότι η κατηγοροποίηση θα επηρεάζεται από αντικείμενα που δε σχετίζονται και δεν μοιάζουν με το δείγμα ελέγχου και θα παράγονται λανθασμένα συμπεράσματα.
- Ένας άλλος τρόπος κατά τον οποίον μπορούν να προκύψουν λανθασμένα συμπεράσματα από τον αλγόριθμο  $k$ -NN είναι η εισαγγή δεδομένων σε αυτόν χωρίς την κατάλληλη και απαραίτητη προεπεξεργασία τους. Έχουμε αναφέρει πως ο αλγόριθμος για να προσφέρει αποτελέσματα χρειάζεται ένα σύνολο εκπαίδευσης, δηλαδή ένα σύνολο αντικειμένων. Τα αντικείμενα αυτά περιγράφονται πλήρως από ένα διάνυσμα στον  $d$ -διάστατο χώρο. Οι  $d$  τιμές του διανύσματος αποτελούν τα χαρακτηριστικά του αντικειμένου. Ωστόσο, δεν έχουν όλα τα χαρακτηριστικά την ίδια φύση και συνεπώς δεν λαμβάνουν τιμές από το ίδιο εύρος τιμών. Για παράδειγμα, αν το σύνολο δεδομένων αφορά ανθρώπους και μερικά από τα χαρακτηριστικά των δειγμάτων είναι το ύψος του ανθρώπου και το βάρος του, τότε μπορεί κανείς να συμπεράνει ότι το βάρος θα λαμβάνει τιμές από ένα μεγαλύτερο εύρος τιμών σε σχέση με το ύψος. Το ύψος, στην περίπτωση που το σύνολο δεδομένων αφορά ενήλικες, μπορεί να κυμαίνεται μεταξύ 150 cm και 200 cm, ενώ το βάρος μπορεί να κυμαίνεται σε τιμές από 40 κιλά έως και 150 κιλά. Αν δε ληφθεί υπ' όψιν το εύρος τιμών των χαρακτηριστικών και δεν κανονικοποιηθεί ώστε όλα τα χαρακτηριστικά να λαμβάνουν τιμές από το ίδιο πεδίο ορισμού με κατάλληλο τρόπο, τότε το μέτρο ομοιότητας θα παράγει λανθασμένα αποτελέσματα, διότι θα μεροληπτεί ως προς ορισμένα χαρακτηριστικά και θα αδιαφορεί για άλλα με βάση τις τιμές που λαμβάνουν. Έτσι, η χλίμακα των χαρακτηριστικών θα επηρεάσει το αποτέλεσμα του αλγορίθμου χωρίς καν να το αντιληφθούμε, γεγονός που δεν πρέπει να συμβεί, καθώς τα χαρακτηριστικά δεν ταξινομούνται σε σειρά σημαντικότητας με βάση το πεδίο τιμών τους.
- Από τον αλγόριθμο των  $k$  πλησιέστερων γειτόνων προκύπτουν αυθαίρετα σχηματισμένα όρια απόφασης. Πιο συγκεκριμένα τα όρια απόφασης μεταβάλλονται πολύ εύκολα, διότι εξαρτώνται από τη μορφή του συνόλου εκπαίδευσης. Αν το  $k$  αυξηθεί, τότε η μεταβλητότητα αυτή των ορίων απόφασης μπορεί να περιοριστεί σημαντικά.
- Ο αλγόριθμος είναι ευαίσθητος σε θορύβους που μπορεί να περιέχονται στο σύνολο εκπαίδευσης.

Αντικείμενα τα οποία μπορεί να αποτελούν ακραίες περπτώσεις (outliers) και δείγματα των οποίων το διάνυσμα χαρακτηριστικών δεν είναι πλήρως συμπληρωμένο "απορρυθμίζουν" τον k-NN και τον οδηγούν στην παραγωγή λάθος προβλέψεων. Συνεπώς και εδώ χρειάζεται μία κατάλληλη προεπεξεργασία των δεδομένων.

## Algorithm

Όσα αναφέρονται παραπάνω για τον k-NN μπορούν να συνοψιστούν στον παρακάτω αλγόριθμο.

---

### Algorithm 1: $\chi$ -Πλησιέστεροι Γείτονες

---

**Result:** Κατηγοριποίηση των δειγμάτων ελέγχου

**for**  $Z = (x', l')$  **do**

    Τυπολόγιση την απόσταση του  $Z$  με τα αντικείμενα του συνόλου εκπαίδευσης ;

    Επέλεξε τα  $k$  πλησιέστερα αντικείμενα ;

    Επέλεξε την ετικέτα του αντικειμένου με Majority Vote ;

**end**

---

## Implementation

Η υλοποίηση του παραπάνω αλγορίθμου έγινε σε python, χωρίς τη χρήση καμίας σχετικής βιβλιοθήκης της γλώσσας. Όλες οι απαραίτητες λειτουργίες του αλγορίθμου οργανώθηκαν σε μία κάλαση. Ο κώδικας εντοπίζεται στο αρχείο exercise1.py.

```

class KNearestNeighbor:
    def __init__(self, train, k=5, accuracy=None):
        self.k = k
        self.train = train
    # Euclidean distance between 2 vectors
    def get_distance(self, data1, data2):
        distance = 0
        data1 = data1[:len(data1) - 1]
        for idx, val in enumerate(data1):
            distance += pow(data1[idx] - data2[idx], 2)
        return math.sqrt(distance)
    # getting neighbours
    def get_neighbours(self, test_instance):
        distances = [self.get_tuple_distance(
            training_instance, test_instance) for training_instance in
        self.train]
        sorted_distances = sorted(distances, key=itemgetter(1))
        sorted_training_instances = [tuple[0] for tuple in sorted_distances]
        return sorted_training_instances[:self.k]
    # private method to convert to tuple instance and its distance
    def get_tuple_distance(self, training_instance, test_instance):

```

```

    return (training_instance, self.get_distance(test_instance,
training_instance))                                25

# getting majority vote (selects category with the most votes) 26
def get_majority_vote(self, neighbours):            27
    categories = [neighbour[-1] for neighbour in neighbours] 28
    count = Counter(categories)                         29
    return count.most_common()[0][0]                   30

# predicting                                              31
def get_predict(self, test_instances):               32
    self.train = self.normalization(self.train)        33
    test_instances = self.normalization(test_instances) 34
    plot_2D_points(self.train, test_instances, labels_color) 35
    predictions = []                                 36
    for x in range(len(test_instances)):              37
        neighbours =                                38
        self.get_neighbours(test_instance=test_instances[x]) 39
        majority_vote = self.get_majority_vote(neighbours) 40
        predictions.append(majority_vote)
    return predictions                                41

# calculate accuracy of knn algorithm                42
def accuracy_calc(self, test_fold, prediction):     43
    labels_test_fold = [record[-1]                  44
                        for record in test_fold] # labels of test set
    successful_prediction = 0                         45
    for idx, val in enumerate(labels_test_fold):      46
        if labels_test_fold[idx] == prediction[idx]:   47
            successful_prediction += 1
    self.accuracy = successful_prediction / \
        float(len(labels_test_fold)) * 100.0          48

# normalization routine for train and test dataset 49
def normalization(self, instances):                 50
    min_feature = []                                51
    max_feature = []                                52
    for i, val in enumerate(instances[0]):           53
        column = []                                54
        for feature in instances:                   55
            column.append(feature[i])
        min_feature.append(min(column))
        max_feature.append(max(column))
    for row in instances:                           56
        for i, val in enumerate(row):                57
            num = float(row[i] - min_feature[i])
            den = float(max_feature[i] - min_feature[i])
            row[i] = num / den
    return instances                                  58

def get_params(self):                            59
    return {"train": self.train, "k": self.k, "accuracy": self.accuracy} 60

```

Η απόσταση μεταξύ των δειγμάτων ελέγχου και εκπαίδευσης γίνεται σύμφωνα με την Ευκλείδια απόσταση, η οποία υλοποιήθηκε όπως φαίνεται παραπάνω στη συνάρτηση `get_distance`. Η αποστάσεις που υπολογίζονται ταξινομούνται και επιλέγονται μόνο οι  $k$  καλύτερες, δηλαδή αυτές που είναι μικρότερες, άρα και τα αντίστοιχα αντικείμενα που είναι πιο κοντά στο δείγμα ελέγχου. Η λειτουργία αυτή υλοποιείται στις συναρτήσεις `get_neighbours` και `get_tuple_distance`. Η τελική επιλογή `label` γίνεται με **majority voting**, όπως περιγράψαμε και στο θεωρητικό μέρος παραπάνω και η αντίστοιχη υλοποίηση βρίσκεται στη ρουτίνα `get_majority_vote`. Επίσης, έχει υλοποιηθεί και μία συνάρτηση `accuracy_calc` που υπολογίζει την ακρίβεια των αποτελεσμάτων του αλγορίθμου. Τέλος, έχουμε υλοποιήση τη συνάρτηση `normalization`, που κανονικοποιεί τα δεδομένα. Η κανονικοποίηση είναι ιδιαιτέρως σημαντική, καθώς τα χαρακτηριστικά πρέπει να έχουν το ίδιο εύρος τιμών. Σε περίπτωση που ένα χαρακτηριστικό λαμβάνει τιμές από το διάστημα  $[0, 1]$  και ένα δεύτερο χαρακτηριστικό από το διάστημα  $[0, 100]$ , το δεύτερο χαρακτηριστικό θα θεωρηθεί από τον αλγόριθμο, που συγκρίνει τα δεδομένα με βάση την Ευκλείδια απόσταση, πιο σημαντικό.

Για να μπορέσει κανείς να στήσει τον κατηγοριοποιητή, πρέπει αρχικά να καλέσει τον `constructor` της `κλάσης` με κατάλληλο `train dataset` και επιλογή  $k$  και έπειτα να καλέσει τη συνάρτηση `get_predict` με όρισμα το επιθυμητό `test dataset`, η οποία θα αποφανθεί για την κατηγορία κάθε `test instance`.

Στο αρχείο `exercise1.py` παραθέτουμε και μία ενδεικτική κλήση του αλγορίθμου για κάποια απλά `train` και `test datasets`, μόνο για τους σκοπούς επίδειξης της λειτουργίας του αλγορίθμου. Τα αντικείμενα διαθέτουν μόνο δύο χαρακτηριστικά, για να είναι εύκολη η οπτικοποίηση του συγκεκριμένου παραδείγματος. Μία εκτενέστερη χρήση του αλγορίθμου γίνεται στην άσκηση 5 της ίδιας εργασίας, όπου και θα αναφερθούμε εκεί αναλυτικά για τα αποτελέσματα του αλγορίθμου.

## Running the code

```
$ python -m venv .env
$ source .env/bin/activate
$ pip install -r requirements.txt
$ python exercise1.py
```

Figure 1: Οδηγίες εκτέλεσης προγράμματος

## Example Usage

Η οπτικοποίηση του απλού και μικρού `train dataset` πριν την κανονικοποίηση δίνεται παρακάτω. Για τις δύο κατηγορίες έχουν χρησιμοποιηθεί τα χρώματα μπλε και πράσινο. Ο αλγόριθμος κατηγοριοποιεί τα δύο κόκκινα σημεία. Το ένα ανήκει στην κατηγορία των πράσινων αντικειμένων και το άλλο στην κατηγορία των μπλε αντικειμένων.

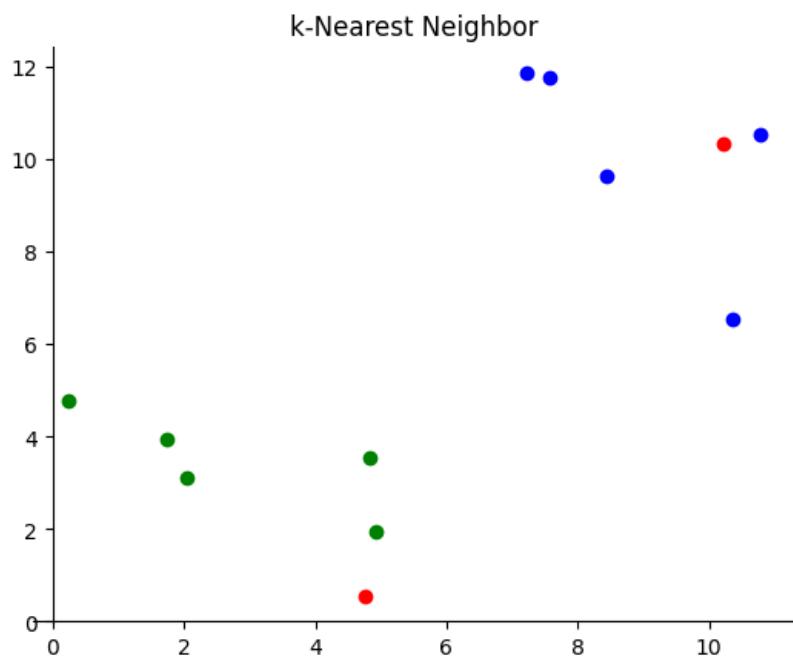


Figure 2: Απλό παράδειγμα κατηγοριοποίησης με k-NN.

## 2. Using the code provided at the class (or implementing your own) explain what is the curse of dimensionality and how is related to the k-NN algorithm.

Είναι συχνό φαινόμενο, σε επίπεδο εκμάθησης αλγορίθμων μηχανικής μάθησης ή εξόρυξης δεδομένων, να περιορίζεται η μελέτη σε προβλήματα που ορίζονται πάνω σε λίγες διαστάσεις, Συνήθως μελετώνται προβλήματα στις 2 ή στις 3 διαστάσεις και αυτό είναι εύλογο, διότι μπορούμε πολύ εύκολα να αναπαραστήσουμε οπτικά τα αποτελέσματα των αλγορίθμων. Ωστόσο, στην πραγματικότητα τα προβλήματα ορίζονται σε περισσότερες διαστάσεις από 2 ή 3.

Οι διαστάσεις σε ένα πρόβλημα ανάλυσης δεδομένων αντικατοπτρίζουν τα χαρακτηριστικά των αντικειμένων που χρησιμοποιούνται ως βάση για την εκπαίδευση του εκάστοτε αλγορίθμου και ως στόχοι του. Για να γίνει πιο εύκολα κατανοητή η έννοια των χαρακτηριστικών θεωρούμε το εξής παράδειγμα: έστω ότι τα αντικείμενα που επεξεργάζεται ο αλγόριθμος αναπαριστούν τα οχήματα: 'αυτοκίνητο' και 'ποδήλατο' και επιθυμούμε να αποφανθούμε για ένα νέο αντικείμενο αν ανήκει στην κλάση των αυτοκινήτων ή των ποδηλάτων. Τα χαρακτηριστικά που μπορούν να χρησιμοποιηθούν για την περιγραφή των αντικείμενων μπορούν ενδεικτικά να είναι, το πλήθος των τροχών, το μέγεθος, το βάρος κ.α.

Φαινεταί λογικό πως όσα περισσότερα χαρακτηριστικά κατέχει το κάθε αντικείμενο, τόσο πιο εύκολο θα είναι για τον αλγόριθμο να προβλέψει σε ποιά κλάση αντικειμένων ανήκει το προς εξέταση αντικείμενο. Ωστόσο, δεν πρέπει να αποχρυφεί ότι ένα πολύ σημαντικό ζήτημα στην περίπτωση των πολλών χαρακτηριστικών (ή αλλιώς πολλών διαστάσεων) είναι οι μεγάλες απαιτήσεις σε μνήμη, υπολογιστική ισχύ και χρόνο. Πέρα από την ανάκυψη αυτού του πρακτικού ζητήματος, οι πολλές διαστάσεις οδηγούν και σε ένα πολύ σημαντικό πρόβλημα που καλείται "η κατάρα των πολλών διαστάσεων" ή αλλιώς "η κατάρα της διαστατικότητας" (The Curse of Dimensionality). Ο όρος προέκυψε το 1961 από τον Richard E. Bellman.

Η κατάρα των πολλών διαστάσεων δεν είναι ένα πρόβλημα που μπορεί να οριστεί αυστηρά σε κάθε αλγόριθμο ανάλυσης δεδομένων. Αυτό συμβαίνει διότι εξαρτάται άμεσα από το σύνολο των δεδομένων και τον εκάστοτε αλγόριθμο. Πιο συγκεκριμένα, το ίδιο μοντέλο μπορεί να παρουσιάζει αρκετά καλή συμπεριφορά, δηλαδή ο αλγόριθμος να δίνει καλό Accuracy, για ένα σύνολο δεδομένων όπου η διάστασή του είναι  $n$  και για μεγαλύτερη διάσταση από αυτήν το Accuracy να μειώνεται σημαντικά. Να σημειώσουμε εδώ ότι το Accuracy χρησιμοποιείται στους αλγορίθμους μηχανικής μάθησης ως μετρική για την αξιολόγηση του εκάστοτε αλγορίθμου. και δίνεται από τον τύπο

$$\text{Accuracy} = \frac{\text{\# correct predictions}}{\text{\# predictions}}.$$

Ουσιαστικά το  $n$  είναι ένα threshold για τον συγκεκριμένο αλγόριθμο και τα συγκεκριμένα δεδομένα. Το threshold αυτό δεν είναι ένα σταθερό μέγεθος και μεταβάλεται αναλόγως τον αλγόριθμο και τα δεδομένα.

Η κατάρα των πολλών διαστάσεων σίγουρα αντιβαίνει στη διαίσθηση που έχει ο άνθρωπος. Θα πίστευε κανείς πως όσα περισσότερα χαρακτηριστικά έχει στη διάθεσή του, τόσο πιο εύκολη θα ήταν η κατηγοριοποίηση των αντικειμένων. Κάτι τέτοιο όμως δε συμβαίνει στην πραγματικότητα. Για να εξηγήσουμε το φαινόμενο αυτό αρχικά θα το περιγράψουμε διαισθητικά και από τη γεωμετρική

σκοπιά των πραγμάτων και στη συνέχεια θα το ορίσουμε με αυστηρά μαθηματικά.

Για τη γεωμετρική προσέγγιση του ζητήματος επιστρατεύουμε ξανά το παράδειγμα με τα οχήματα που προαναφέραμε. Για να αποφανθούμε για την κατηγορία οχήματος για κάποιο αντικείμενο που δε γνωρίζουμε την κατηγορία του μία απλή τεχνική είναι να διαμερίσουμε τον χώρο στον οποίο ζουν τα δεδομένα και το καινούριο αντικείμενο να λάβει *label* ανάλογα με την περιοχή του χώρου στην οποία εντοπίζεται. Αν δηλαδή βρίσκεται σε περιοχή που υπερισχύει η κλάση των ποδηλάτων, δηλαδή σε αυτήν την περιοχή εντοπίζονται περισσότερα ποδήλατα, τότε θα λάβει το *label* "ποδήλατο" και αντίστοιχα για οποιαδήποτε άλλη κατηγορία οχήματος.

Αν αποφασίσουμε τα αντικείμενα να περιγράφονται από ένα χαρακτηριστικό, τότε τα σημεία που αναπαριστούν τα οχήματα τοποθετούνται πάνω σε μία ευθεία. Αν διαιρέσουμε την ευθεία σε ίσα τμήματα, παράγονται  $m$  το πλήθος τέτοια τμήματα. Αν ακολουθήσουμε την ίδια διαδικασία στις 2 διαστάσεις, δηλαδή αν τα αντικείμενα μας τώρα περιγράφονται από 2 χαρακτηριστικά, παρατηρούμε ότι τα κελιά που δημιουργούνται, τα οποία έχουν τη μορφή τετραγώνων είναι πολλά περισσότερα σε αριθμό, ενώ στις 3 διαστάσεις η ίδια διαδικασία θα παράξει ίσους κύβους ως κελιά και το πλήθος αυτών θα είναι πολύ μεγαλύτερο σε σχέση με το πλήθος των κελιών στις 2 διαστάσεις και στη 1 διάσταση.

πλήθος κελιών στη 1 διάσταση  $\ll$  πλήθος κελιών στις 2 διαστάσεις  $\ll$  πλήθος κελιών στις 3 διαστάσεις

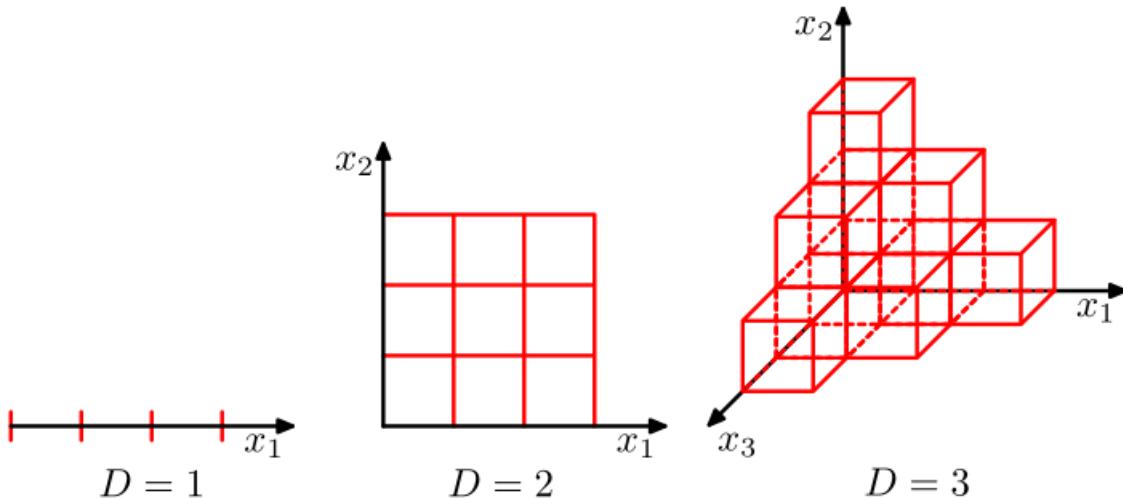


Figure 3: Διάσπαση του χώρου αναπαράστασης των δεδομένων σε ίσα κελιά, για χώρο διάστασης  $D = 1$ ,  $D = 2$  και  $D = 3$ . Το πλήθος των κελιών αυξάνεται σημαντικά όσο αυξάνεται η διάσταση του χώρου. (πηγή: Pattern Recognition and Machine Learning, Christopher M. Bishop)

Να επισημάνουμε στο σημείο αυτό, ότι ο όρος "κελί" εδώ χρησιμοποιείται καταχρηστικά και αναφέρεται είτε σε ευθύγραμμο τμήμα, είτε σε τετράγωνο, είτε σε κύβο αναλόγως με το  $D$ .

Μπορεί κανείς εύκολα να συμπεράνει πώς όσο αυξάνεται το  $D$ , δηλαδή το πλήθος των χαρακτηριστικών, τόσα περισσότερα κελιά θα μένουν χωρίς σημεία. Καθώς το πλήθος των δεδομένων είναι

σταυθερό και καθορισμένο και για τις τρεις διαφορετικές περιπτώσεις D, το ίδιο πλήθος σημείων αντιστοιχεί σε περισσότερα κελιά καθώς το D αυξάνεται. Συνεπώς, όσο αυξάνεται η διάσταση του προβλήματος τόσο τα σημεία διασπείρονται στο χώρο και οι μεταξύ τους αποστάσεις μεγαλώνουν σημαντικά. Όσο το πλήθος των διαστάσεων αυξάνεται, τόσο τα δεδομένα τοποθετούνται πιο σποραδικά στον χώρο που καταλαμβάνουν.

Η παρατήρηση αυτή μπορεί να επιβεβαιωθεί και πειραματικά. Επιλέξαμε να τρέξουμε τον κώδικα που αφορά την κατάρα της διαστατικότητας που παρουσιάστηκε στο μάθημα και να τον παραμετροποιήσουμε με τέτοιον τρόπο, ώστε να μπορέσουμε να παράξουμε ασφαλή συμπεράσματα σχετικά με το φαινόμενο. Παραθέτουμε παραχάτω τον κώδικα για λόγους διευκόλυνσης του αναγνώστη.

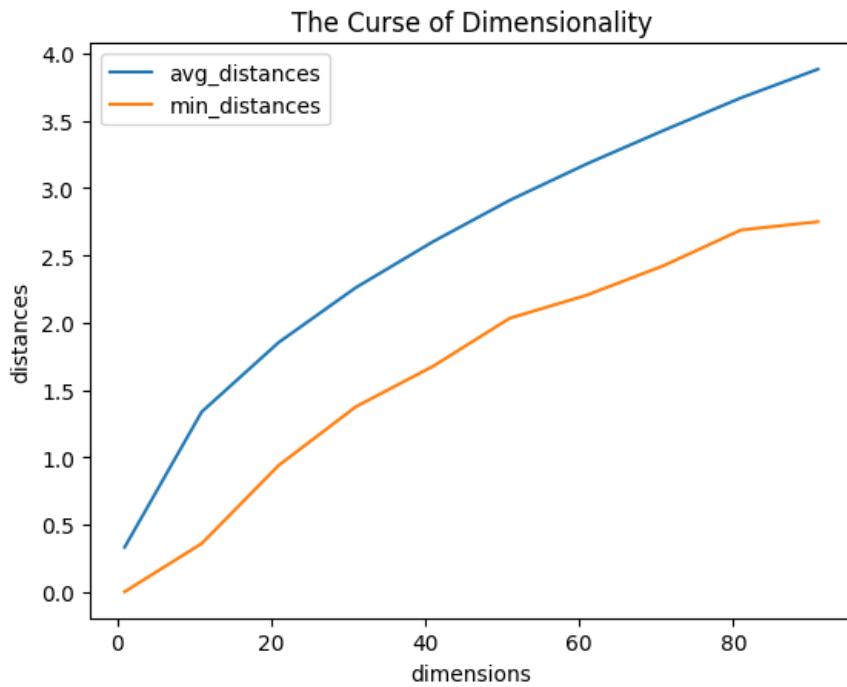
```

1  def random_point(dim):
2      return [random.random() for _ in range(dim)]
3
4  def random_distances(dim, num_pairs):
5      return [distance(random_point(dim), random_point(dim))
6              for _ in range(num_pairs)]
7
8  dimensions = range(1, 10000, 1000)
9
10 avg_distances = []
11 min_distances = []
12
13 random.seed(0)
14
15 for dim in dimensions:
16     distances = random_distances(dim, 10000) # 10,000 random pairs
17     avg_distances.append(mean(distances)) # track the average
18     min_distances.append(min(distances)) # track the minimum
19     print(f"DIMENSION: {dim} MIN DISTANCE {min(distances)} MEAN DISTANCE:
20 {mean(distances)}")
21
22 plt.title('The Curse of Dimensionality')
23
24 plt.plot(dimensions, avg_distances, label = "avg_distances")
25 plt.plot(dimensions, min_distances, label = "min_distances")
26
27 # naming the x axis
28 plt.xlabel('dimensions')
29 # naming the y axis
30 plt.ylabel('distances')
31
32 plt.legend()
33 plt.show()

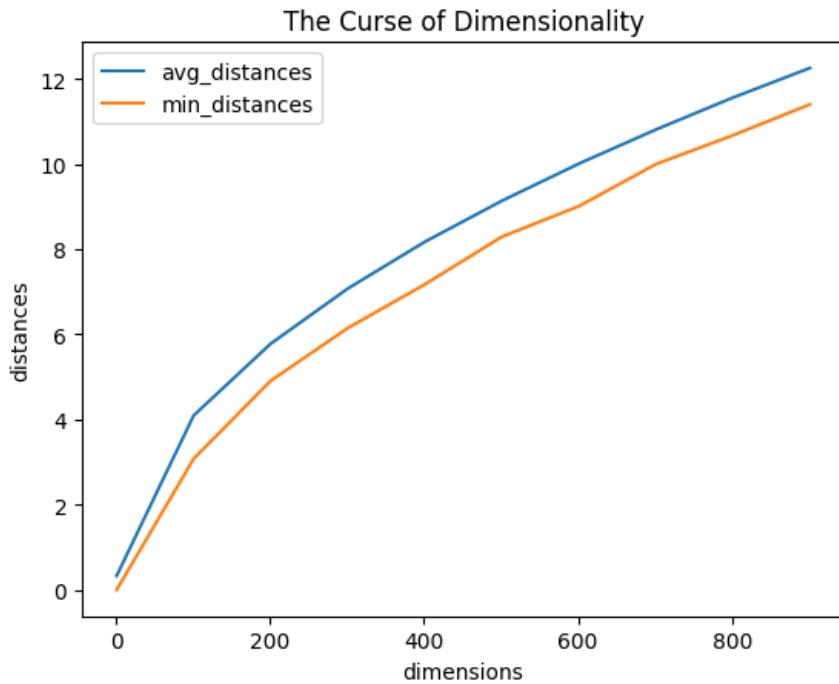
```

---

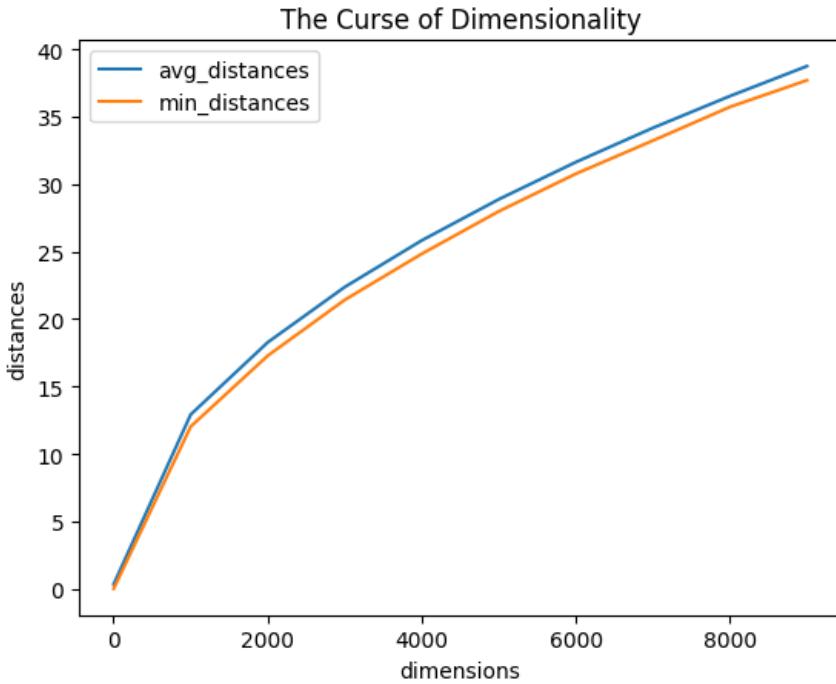
Τρέξαμε τον κώδικα για διαφορετικό εύρος διαστάσεων και πλήθος σημείων. Ενδεικτικά αποτελέσματα παρουσιάζονται στα διαγράμματα παραχάτω.



**Figure 4:** Η κατάρα της διαστατικότητας: αναπαράσταση της μέσης και της ελάχιστης απόστασης σημειών σε χώρους διάστασης από  $D = 1$  εως  $D = 100$  για σύνολο δεδομένων 10000 σημείων



**Figure 5:** Η κατάρα της διαστατικότητας: αναπαράσταση της μέσης και της ελάχιστης απόστασης σημειών σε χώρους διάστασης από  $D = 1$  εως  $D = 1000$  για σύνολο δεδομένων 10000 σημείων



**Figure 6:** Η κατάρα της διαστατικότητας: αναπαράσταση της μέσης και της ελάχιστης απόστασης σημείων σε χώρους διάστασης από  $D = 1$  εως  $D = 10000$  για σύνολο δεδομένων 10000 σημείων

Παρατηρούμε ότι όσο αυξάνεται το πλήθος των χαρακτηριστηκών, τόσο αυξάνεται και η απόσταση των σημείων μεταξύ τους και μάλιστα με πολύ γρήγορο ρυθμό, γεγονός που πιστοποιεί όσα ήδη έχουμε αναφέρει για την κατάρα των πολλών διαστάσεων. Η αύξηση της μέσης απόστασης μεταξύ των σημείων τείνει να εκφυλίσσει την εννοια της απόστασης, πλέον το πόσο απέχουν δύο αντικείμενα δεν επαρκεί για να παραχθούν ασφαλή συμπεράσματα. Όσο μεγαλώνει η διάσταση, τόσο αραιώνει ο χώρος από σημεία που αναπαριστούν δεδομένα και αυτό μπορεί να οδηγήσει στη σκέψη ότι τα σημεία αυτά μπορεί να μην είναι πλέον αντιπροσωπευτικά του προβλήματος και να μην πρόκειται για σημεία που αναπαριστούν τον "χανόνα", αλλά την εξαίρεση, γεγονός που δυσκολεύει την παραγωγή ορθών συμπερασμάτων.

Αφού κατανοήσαμε διαισθητικά και πειραματικά την κατάρας της διαστατικότητας μπορούμε πλέον να περάσουμε στον πιο αυστηρό και με μαθηματικό τρόπο ορισμό του φαινομένου.

Αρχικά, πρέπει να αναφέρουμε ορισμένες σημαντικές μαθηματικές έννοιες, για να μπορέσουμε να ορίσουμε το φαινόμενο με αυστηρό τρόπο. Βασιζόμαστε στην ανάλυση και στον συμβολισμό που παρουσιάζεται στο [1]

Στην περίπτωση του 1-NN, θεωρούμε ότι τα σημεία που αποτελούν το σύνολο εκπαίδευσης ανήκουν στον χώρο  $X$ , για τον οποίον ισχύει χωρίς βλάβη της γενικότητας ότι

$$X = [0, 1]^d$$

, όπου  $d$  είναι η διάσταση του χώρου ( $d = 1, 2, \dots$ ). Ο χώρος  $X$  είναι εξοπλισμένος με μία μετρική  $\rho$ , για την οποία ισχύει

$$\rho : X \times X \rightarrow \mathbb{R}$$

και η οποία ακολουθεί μερικούς κανόνες:

- $\rho(x, x') \geq 0 \quad \forall x, x'$  Θετικότητα (Positivity)
- $\rho(x, x') = 0 \Leftrightarrow x = x'$  Θετικότητα (Positivity)
- $\rho(x, x') = \rho(x', x)$  Συμμετρία (Symmetry)
- $\rho(x, x') \leq \rho(x, x'') + \rho(x'', x') \quad \forall x, x', x''$  Τριγωνική Ανισότητα (Triangle Inequality)

Μία τέτοια μετρική είναι η Ευκλείδια απόσταση:

$$\rho(x, x') = \|x - x'\| = \sqrt{\sum_{i=1}^d (x_i - x'_i)^2}$$

Τιοθετούμε την ευκλείδια απόσταση ως μετρική σε αυτήν την περίπτωση. Τα *labels*, δηλαδή οι κατηγορίες που αποδίδονται σε κάθε αντικείμενο ανήκουν στον χώρο  $Y$ , για τον οποίο ισχύει χωρίς βλάβη της γενικότητας ότι

$$Y = [0, 1]$$

Θεωρούμε σύνολο  $S$  τέτοιο ώστε

$$S = \{(x_1, y_1), \dots, (x_m, y_m)\}$$

Με  $m$  συμβολίζεται το πλήθος των αντικειμένων του συνόλου εκπαίδευσης  $S$  του αλγορίθμου. Έστω  $D$  μία κατανομή στο  $X \times Y$  και η μία συνάρτηση για την οποία ισχύει ότι

$$\eta : \mathbb{R}^d \rightarrow \mathbb{R}$$

και εκφράζει την πιθανότητα

$$\eta(x) = P[y = 1|x]$$

Η  $\eta$  είναι  $c$ -Lipschitz συνάρτηση, δηλαδή ισχύει ότι υπάρχει κάποια σταθερά  $c$ , τέτοια ώστε

$$|\eta(x) - \eta(x')| \leq c \|x - x'\|$$

Αν με  $h_s$  συμβολίζουμε τον κανόνα που έχουμε επιλέξει για την αξιολόγηση των σημείων, με  $h_s^*$  τον βέλτιστο κανόνα αξιολόγησης σημείων και με  $L_D$  το λάθος ενός κανόνα, είτε αυτός είναι ο  $h$ , είτε ο  $h^*$ , τότε το αναμενόμενο λάθος στον 1-NN αλγόριθμο φράσσεται ως εξής:

$$E[L_D(h_s)] \leq 2L_D(h^*) + 4c \sqrt{d} m^{-\frac{1}{d+1}}$$

Στην περίπτωση όπου  $k \geq 2$ , το αντίστοιχο άνω φράγμα γίνεται

$$E[L_D(h_s)] \leq (1 + \sqrt{\frac{8}{k}})L_D(h^*) + (6c\sqrt{d} + k)m^{-\frac{1}{d+1}}$$

Συνεπώς, το σφάλμα που παράγεται από τον αλγόριθμο (ανεξάρτητα από την επιλογή του  $k$ ) εξαρτάται από τις ιδιότητες της κατανομής και το  $m$ , δηλαδή το πλήθος των αντικειμένων εκπαίδευσης του αλγορίθμου. Ιδανικά, το σφάλμα θέλουμε να είναι μικρότερο από μία μικρή ποσότητα  $\varepsilon$ . Για να συμβεί αυτό πρέπει

$$E[L_D(h_s)] \leq \varepsilon$$

Στην περίπτωση του 1-NN

$$\begin{aligned} 4c\sqrt{d}m^{-\frac{1}{d+1}} \leq \varepsilon &\Leftrightarrow \\ m^{-\frac{1}{d+1}} \leq \frac{\varepsilon}{4c\sqrt{d}} &\Leftrightarrow \\ \frac{1}{m^{\frac{1}{d+1}}} \leq \frac{\varepsilon}{4c\sqrt{d}} &\Leftrightarrow \\ m^{\frac{1}{d+1}} \geq \frac{4c\sqrt{d}}{\varepsilon} &\Leftrightarrow \\ \sqrt[d+1]{m} \geq \frac{4c\sqrt{d}}{\varepsilon} &\Leftrightarrow \\ m \geq \frac{4c\sqrt{d}^{d+1}}{\varepsilon} & \end{aligned}$$

Στην περίπτωση του  $k$ -NN, με  $k \geq 2$

$$\begin{aligned} (6c\sqrt{d} + k)m^{-\frac{1}{d+1}} \leq \varepsilon &\Leftrightarrow \\ m^{-\frac{1}{d+1}} \leq \frac{\varepsilon}{6c\sqrt{d} + k} &\Leftrightarrow \\ \frac{1}{m^{\frac{1}{d+1}}} \leq \frac{\varepsilon}{6c\sqrt{d} + k} &\Leftrightarrow \\ m^{\frac{1}{d+1}} \geq \frac{6c\sqrt{d} + k}{\varepsilon} &\Leftrightarrow \\ \sqrt[d+1]{m} \geq \frac{6c\sqrt{d} + k}{\varepsilon} &\Leftrightarrow \\ m \geq \frac{6c\sqrt{d} + k}{\varepsilon}^{d+1} & \end{aligned}$$

Συνεπώς, το πλήθος των αντικειμένων που απαρτίζουν το σύνολο εκπαίδευσης του  $k$ -NN πρέπει να αυξάνεται εκθετικά με βάση τη διάσταση του χώρου στον οποίον ανήκει το σύνολο των αντικειμένων που εξετάζονται, ώστε ο αλγόριθμος να παράγει ασφαλή αποτελέσματα. Αυτή η εκθετική εξάρτηση του αλγορίθμου από τη διάσταση είναι η κατάρα της διαστατικότητας.

Παραθέτουμε ενδεικτικά στους παραχάτω πίνακες το μέγεθος ενός συνόλου εκπαίδευσης που χρειάζεται ο k-NN, για να παράξει σωστά αποτελέσματα. Το c εδώ είναι 2 και το e ισούται με 0.8

1-NN	
Διάσταση Χώρου Συνόλου Εκπαίδευσης (d)	Πλήθος Σημείων Συνόλου Εκπαίδευσης (m)
1	100
3	83521
5	113379904
7	208827064576
9	5904900000000000

3-NN	
Διάσταση Χώρου Συνόλου Εκπαίδευσης (d)	Πλήθος Σημείων Συνόλου Εκπαίδευσης (m)
1	225
3	331776
5	729000000
7	2251875390625
9	8140406085191601

10-NN	
Διάσταση Χώρου Συνόλου Εκπαίδευσης (d)	Πλήθος Σημείων Συνόλου Εκπαίδευσης (m)
1	576
3	1185921
5	3518743761
7	14048223625216
9	64925062108545024

Ο υπολογισμός των τιμών που παρουσιάζονται στους παραχάτω πίνακες έγινε με βάση το πρόγραμμα που βρίσκεται στο αρχείο exercise2.py.

---

```

import math                                         1
dimensions = range(1, 10, 2)                      2
c = 2                                              3
e = 0.8                                            4
print("1-NN")                                      5
for dim in dimensions:                            6
    frac = int(4 * c * math.sqrt(dim) / e)          7
    points = pow(frac, dim + 1)                      8
    print(f"Dimension: {dim} points: {points}")      9
print("k-NN")                                       10
neighboors = range(2, 11, 1)                       11
for k in neighboors:                                12
    for dim in dimensions:                          13
        frac = int((6 * c * math.sqrt(dim)) + k / e) 14
        points = pow(frac, dim + 1)                  15
        print(f"k = {k} Dimension: {dim} points: {points}") 16

```

---

Θα μπορούσε κανείς να υποστηρίξει ότι η κατάρα της διαστατικότητας μπορεί να αντιμετωπιστεί

με την προσθήκη κάθε φορά όλων των απαραίτητων σημείων, ώστε ο αλγόριθμος να παράγει ασφαλή αποτελέσματα. Ωστόσο, αυτή δεν είναι μία αποδοτική λύση, καθώς το πλήθος των σημείων που θα χρειάζεται κάθε φορά ο αλγόριθμος θα αυξάνεται εκθετικά σύμφωνα με τη διάσταση του χώρου των αντικειμένων. Συνεπώς, μία πιο ασφαλής προσέγγιση του ζητήματος είναι να περιορίσουμε τις διαστάσεις, δηλαδή τα χαρακτηριστικά των αντικειμένων. Δεν είναι πάντοτε όλα τα χαρακτηριστικά σημαντικά για την εξαγωγή συμπερασμάτων στη μηχανική μάθηση.

Σε γενικές γραμμές, η μείωση των πολλών διαστάσεων παρέχει πολλά πλεονεκτήματα. Εκτός από τη συμβολή της στο ζήτημα της κατάρας των πολλών διαστάσεων που μελετάμε εδώ, μπορεί να μειώσει τον ύψορυθο, αφού παραχάμπτοντας ορισμένα μη οφέλιμα για τον αλγόριθμο χαρακτηριστικά, τα συμπεράσματα που θα προκύψουν θα βασίζονται μόνο σε γνωρίσματα που σχετίζονται με το εκάστοτε αντικείμενο μελέτης. Βεβαίως, η μείωση των διαστάσεων συμβάλλει και στην ευκολότερη οπτικοποίηση των δεδομένων, που πολλές φορές είναι σημαντική για την κατανόηση ενός προβλήματος.

Οι διαστάσεις μπορούν να μειωθούν με διάφορους τρόπους, έτσι ώστε ο αλγόριθμος να μην αντιμετωπίζει πρόβλημα ως προς τη διάσταση και τα δεδομένα. Μία επιλογή είναι να συνδυαστούν μερικά από τα χαρακτηριστικά των δειγμάτων και να παράξουν ένα νέο χαρακτηριστικό που να τα συνοψίζει. Σε αυτήν την κατεύθυνση λειτουργούν οι κλασσικές τεχνικές PCA και SVD. Πρόκειται για τεχνικές γραμμικής άλγεβρας, η οποίες προβάλλουν τα δεδομένα από ένα χώρο  $m$  διαστάσεων σε ένα χώρο  $p$  διαστάσεων, όπου  $p < m$ . Πιο αναλυτικά, το PCA (Principal Component Analysis) ή αλλιώς στα ελληνικά η ανάλυση κυρίων συνιστωσών παράγει καινούρια χαρακτηριστικά (συνιστώσες) τα οποία ικανοποιούν ορισμένες συνθήκες.

- Είναι γραμμικοί συνδυασμοί των αρχικών χαρακτηριστικών
- Είναι κάθετα μεταξύ τους
- Λαμβάνουν τη μέγιστη απόκλιση στα δεδομένα

Έτσι, τα αρχικά αντικείμενα περιγράφονται πλέον από ένα μικρότερο πλήθος χαρακτηριστικών και τοποθετούνται σε χώρο μικρότερης διάστασης.

Μία άλλη τεχνική είναι η επιλογή ενός υποσυνόλου χαρακτηριστικών και η διαγραφή των υπολοίπων, που δεν είναι σημαντικά για το υπό μελέτη πρόβλημα. Η τελευταία τεχνική καλείται επιλογή γνωρισμάτων (feature selection). Θα μπορούσε κανείς να ισχυριστεί πως η απαλοιφή χαρακτηριστικών μπορεί να αποβεί μοιραία για τα αποτελέσματα του αλγορίθμου, καθώς σε αυτήν την περίπτωση διακυβεύονται τα ποσά πληροφορίας που έχει ο αλγόριθμος στη διάθεσή του. Ωστόσο, αυτή η πρώτη εκτίμηση δεν είναι και η σωστή, διότι τα δεδομένα μπορεί να περιέχουν χαρακτηριστικά που είναι περιττά ή άσχετα. Με τον όρο περιττά χαρακτηριστικά εννοούμε εκείνα τα χαρακτηριστικά που δεν προσφέρουν περισσότερη πληροφορία, ίσα ίσα διπλασιάζουν ένα μέρος αυτής, καθώς η πληροφορίες που παρέχουν είναι ήδη γνωστές από άλλα χαρακτηριστικά. Με τον όρο άσχετα χαρακτηριστικά αναφερόμαστε σε εκείνα τα γνωρίσματα που δεν περιέχουν καμία πληροφορία από άποψη μηχανικής μάθησης. Έτσι, αυτές οι δύο κατηγορίες χαρακτηριστικών μπορούν να αποβούν μοιραίες για τα αποτελέσματα του εκάστοτε αλγορίθμου και πρέπει να απομακρύνονται. Φυσικά, δεν είναι το ίδιο εύκολη η απομάκρυνση / επιλογή χαρακτηριστικών σε όλες τις περιπτώσεις. Υπάρχουν χαρακτηριστικά (όπως για παράδειγμα ένας αναγνωριστικός αριθμός  $id$ ) που με την κοινή λογική μπορούν να απομακρυνθούν άμεσα από το σύνολο των στιγμιοτύπων, ωστόσο η επιλογή χαρακτηριστικών δεν είναι πάντοτε τόσο απλή υπόθεση. Πρέπει να επιλεχθούν εκείνα τα χαρακτηριστικά που θα βοηθήσουν τον αλγόριθμο να απόδοσει τα μέγιστα. Φυσικά, αυτή δεν είναι μια γνώση που έχουμε άπό πριν όταν μας δίνεται ένα σύνολο δεδομένων, οπότε μία

εύλογη τεχνική θα ήταν να δοκιμάσουμε όλους τους πιθανούς συνδυασμούς χαρακτηριστικών, για να καταλήξουμε στο καταλληλότερο σύνολο δεδομένων. Ωστόσο, αν το πλήθος των χαρακτηριστικών είναι  $n$ , τότε το σύνολο όλων των υποσυνόλων των αντικειμένων είναι  $2^n$ , γεγονός που καθιστά απαγορευτική την απλοϊκή αυτή προσέγγιση στο ζήτημα.

3.

1. Suppose there is a set of points on a two-dimensional plane from two different classes. Points in class Red are  $(0, 1)$ ,  $(2, 3)$ ,  $(4, 4)$  and points in class Blue are  $(2, 0)$ ,  $(5, 2)$ ,  $(6, 3)$ . Draw the k-nearest-neighbor decision boundary for  $k = 1$  as we discussed in the lecture. Experiment yourself with two or more different distance metrics. Present your results.

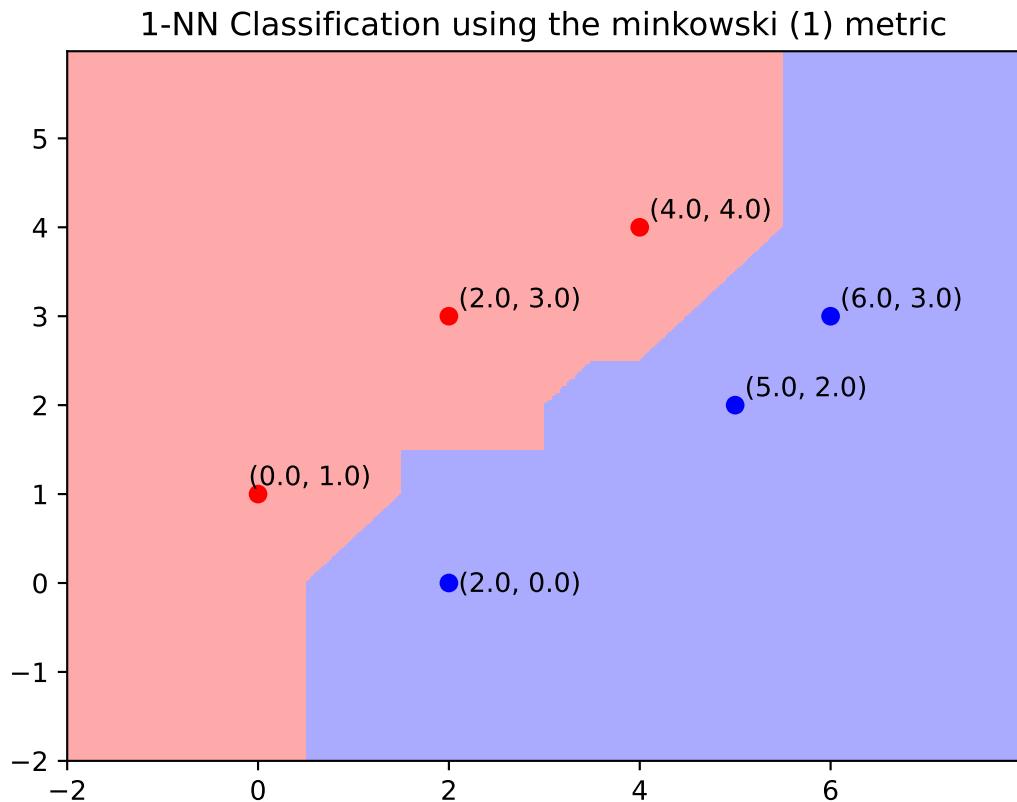


Figure 7: Το όριο απόφασης με χρήση της απόστασης Manhattan

### 1-NN Classification using the minkowski (2) metric

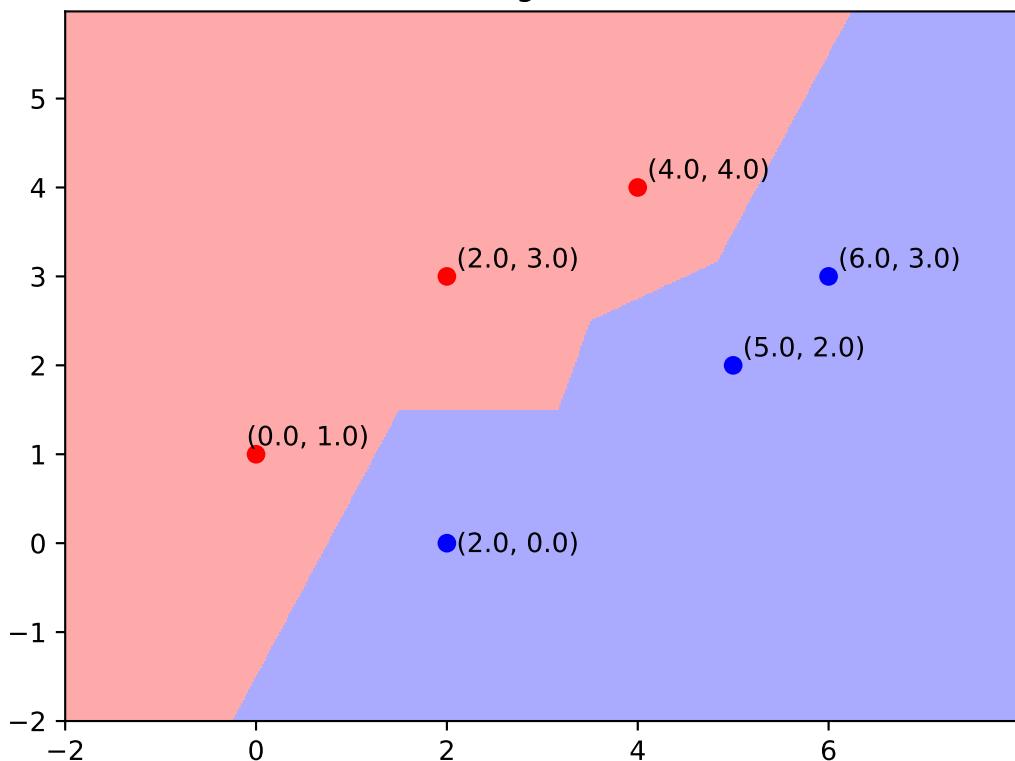


Figure 8: Το όριο απόφασης με χρήση της Ευκλείδιας απόστασης

### 1-NN Classification using the minkowski (3) metric

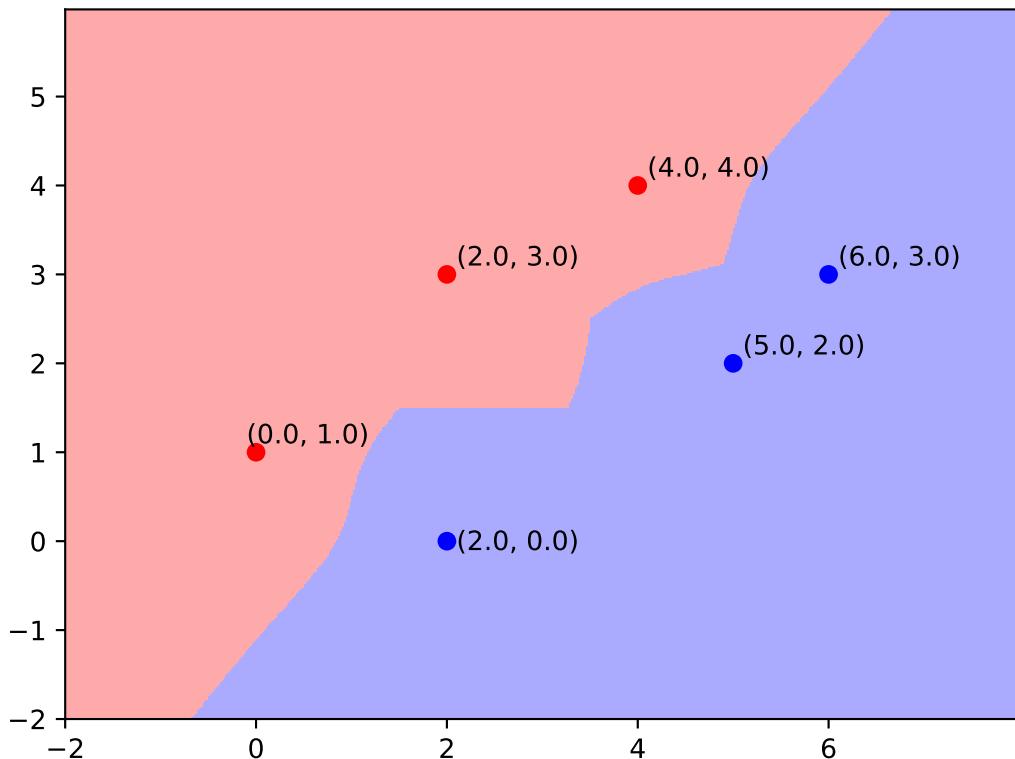


Figure 9: Το όριο απόφασης με χρήση της απόστασης Minkowski(3)

Το όριο απόφασης αλλάζει ανάλογα με τη μετρική που χρησιμοποιείται για την εύρεση των κοντινότερων γειτόνων.

Αυτό ήταν αναμενόμενο καθώς διαφορετικές μετρικές μπορούν να οδηγήσουν στην επολογή διαφορετικών σημείων ως κοντινότερους γείτονες με αποτέλεσμα η κατηγοριοποίηση κάποιου σημείου να οδηγήσει σε διαφορετικό αποτέλεσμα.

**2. If the y-coordinate of each point was multiplied by 5, what would happen to the  $k = 1$  boundary? Draw a new picture. Explain whether this effect might cause problems in practice.**

Αυτό έχει ως αποτέλεσμα σημεία που διαφέρουν ελαφρώς στην θέση ως προς τον οριζόντιο άξονα, να έχουν παροούσιαζουν διαφορετικές κατηγοριοποίησεις. Αυτό αποτελεί έκφραση του υψηλού variance του αλγορίθμου για  $k = 1$ .

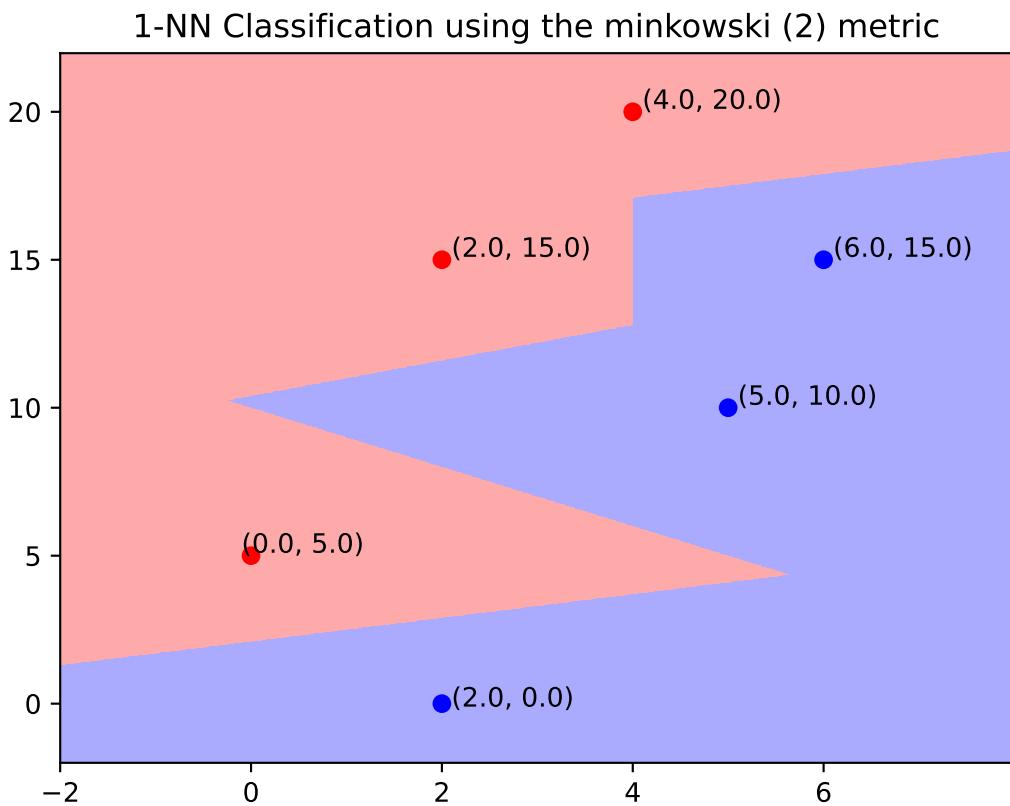


Figure 10: Το όριο απόφασης αν η γ συντεταγμένη κάθε διοθέντος σημείο πενταπλασιαστεί

### 3. Can you draw the decision boundary for k=3 ?

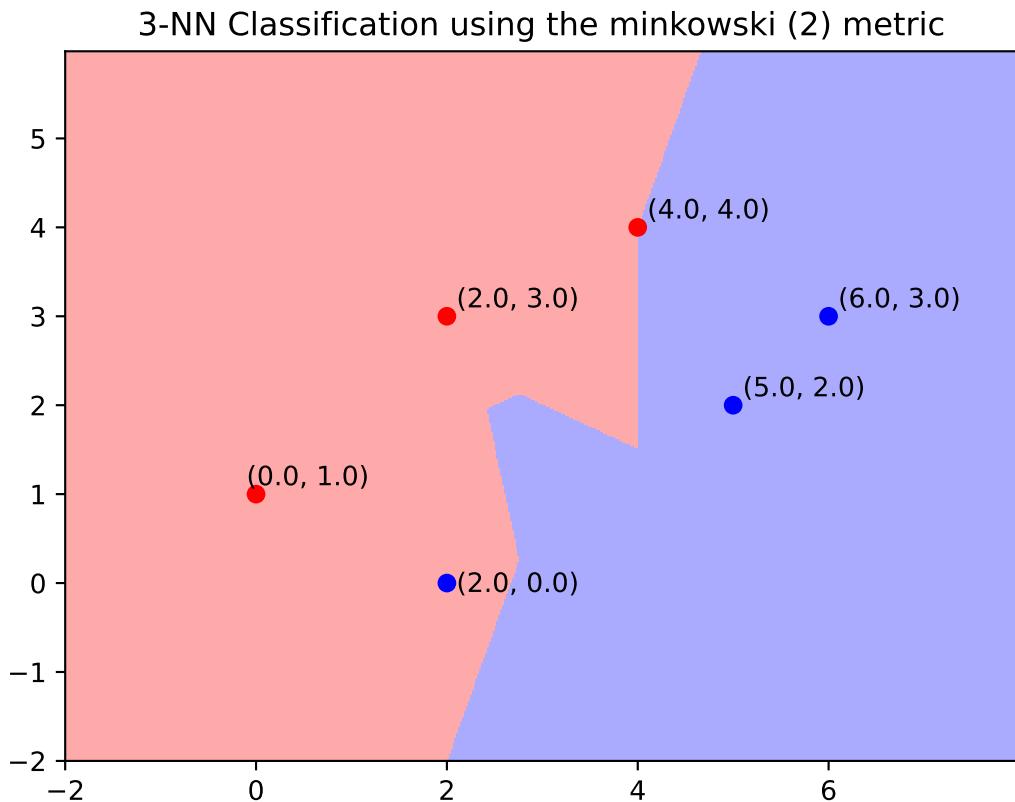


Figure 11: Το όριο απόφασης με χρήση 3 γειτόνων

Όπως βλέπουμε το όριο απόφασης αλλάζει και βάσει του πλήθους των γειτόνων. Αυτό ήταν επίσης αναμενόμενο, καθώς τώρα η κατηγοριοποίηση δεν εξαρτάται μόνο από ένα σημείο αλλά από πολλαπλά.

4. Suppose now we have a test point at  $(1, 2)$ . How would it be classified under 3-NN? Given that you can modify the 3-NN decision boundary by adding points to the training set in the diagram, what is the minimum number of points that you need to add to change the classification at  $(1, 2)$ ? Provide also the coordinates for these new points and justify your answer.

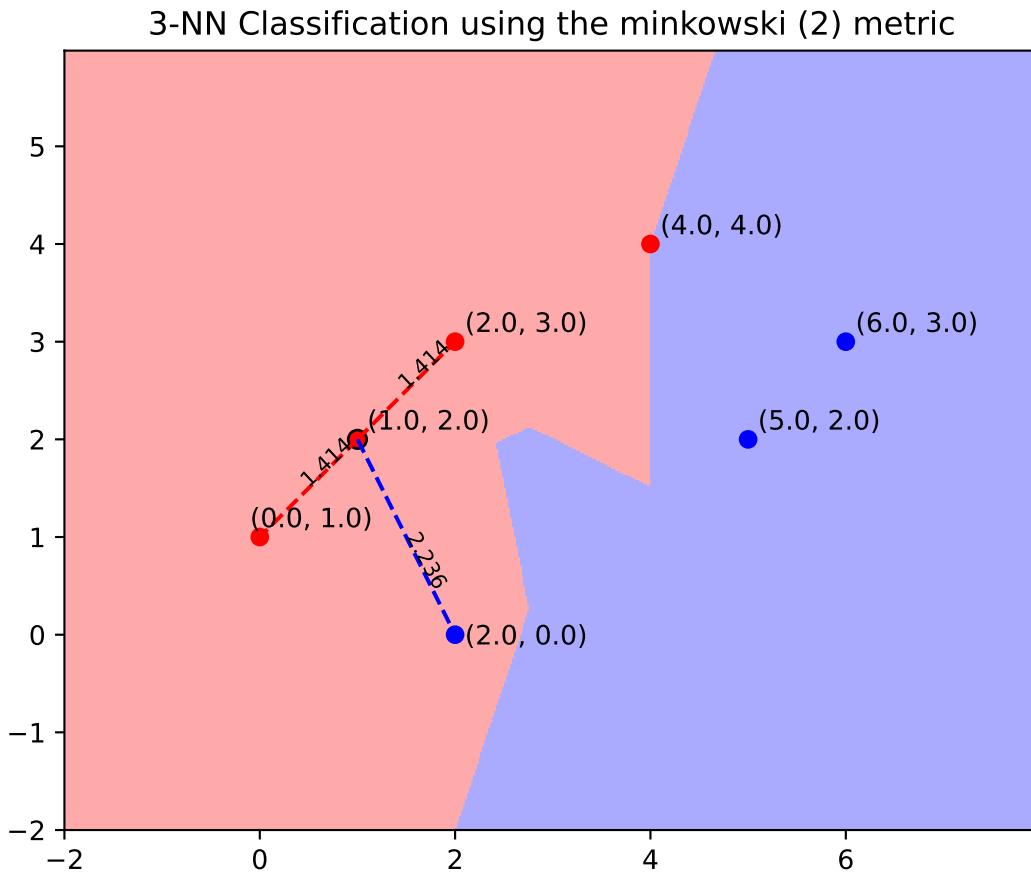


Figure 12: Κατηγοριποίηση του σημείου  $(1, 2)$  με χρήση 3 γειτόνων

Δεδομένου ότι χρησιμοποιούμε 3 γείτονες και τα 3 κοντινότερα σημεία στο σημείο  $(1, 2)$ , είναι τα σημεία  $(0, 1)$ ,  $(2, 3)$ ,  $(2, 0)$ , εκ των οποίων τα δύο κοντινότερα είναι κόκκινα αν θέλουμε να αλλάξουμε το αποτέλεσμα της κατηγοριποίησης, δηλαδή το σημείο  $(1, 2)$  να αντιστοιχιστεί στην μπλέ κατηγορία πρέπει να προσθέσουμε τουλάχιστον 2 μπλέ σημεία σε απόσταση μικρότερη από τα σημεία  $(0, 1)$ ,  $(2, 3)$ .

Προσθέτουμε λοιπόν στο σύνολο εκπαίδεσης τα μπλέ σημεία  $(0, 2)$ ,  $(2, 2)$  και έχουμε το ακόλουθο αποτέλεσμα

### 3-NN Classification using the minkowski (2) metric

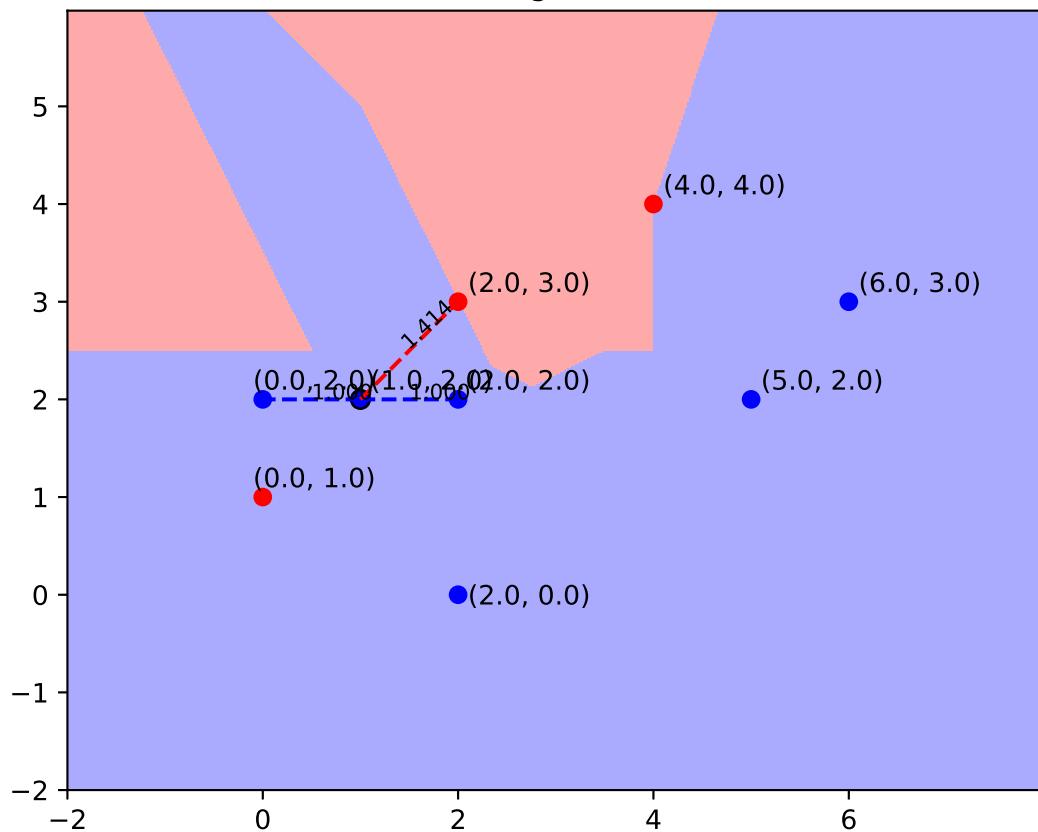


Figure 13: Προσθήκη σημείων στο σύνολο εκμάθησης έτσι ώστε να αλλάξει το αποτέλεσμα της κατηγοριοποίησης του σημείου  $(1, 2)$

```
$ python -m venv .env
$ source .env/Scripts/activate
$ pip install -r requirements.txt
$ python exercise3.py --help
Usage: exercise3.py [OPTIONS]

Demonstrate the differences in decision boundaries in regards to K-NN
classification

Options:
  --neighbors INTEGER          the number of neighbors to use
  --metric TEXT                the distance metric to use
  --p INTEGER                  power parameter for the Minkowski metric
  --step FLOAT                 The step size in the mesh
  --offset FLOAT               the coordinate label offset
  --margin INTEGER              the x, y axis margin
  --fit <FLOAT FLOAT>...      fit the model using 'fit' as training data and
                               'classes' as target values

  --classes INTEGER            fit the model using 'fit' as training data and
                               'classes' as target values

  --predict <FLOAT FLOAT>...   predict the class labels for the provided data.
  --connect                     show connecting line to k nearest neighbors
  --save                        save the resulting figure
  --filename TEXT               where to save the resulting figure
  --help                         Show this message and exit.
```

Figure 14: Running the code

**4. How long does it take for k-NN to classify one point? Or in other words what is the testing complexity for one instance? Assume your data has dimensionality d, you have n training examples and use Euclidean distance. Assume also that you use a quick select implementation which gives you the k smallest elements of a list of length m in O(m).**

Απαιτείται ο υπολογισμός της Ευκλείδιας απόστασης του διοθέντος σημείου από κάθε σημείο εντός του συνόλου μάθησης.

Λαμβάνοντας υπ' όψιν, ότι τα δεδομένα μας είναι d διαστάσεων, ο υπολογισμός της Ευκλείδιας απόστασης μεταξύ δύο σημείων απαιτεί  $O(d)$  χρόνο.

Ως εκ τούτου, για τον υπολογισμό όλων των αποστάσεων απαιτείται  $O(d \cdot n)$  χρόνος.

Έχοντας στη διάθεσή μας μια αποδοτική μέθοδο επιλογής των k μικρότερων στοιχείων μίας λίστας, μπορούμε να βρούμε τις k εκ των n μικρότερες αποστάσεις από το διοθέν σημείο σε χρόνο  $O(n)$ .

Σε αυτό το σημείο πρέπει να αναφέρουμε ότι έχοντας στη διάθεση μας αυτή τη μέθοδο μπορούμε εύκολο να τη σχεδιάσουμε μία εξίσου αποδοτική μέθοδο που επιστρέφει τους k κοντινότερους γείτονες, με χρήση παραδείγματος χάριν ενός πίνακα καταχερματισμού.

Ως εκ τούτου, δεν υπάρχει κάποια επιπλέον χρονική, ασυμπτωτικά τουλάχιστον, επίπτωση στην αποδοτικότητα του αλγορίθμου κατηγοριοποίησης.

Έτσι, η συνολική πολυπλοκότητα κατηγοριοποίησης ενός σημείου είναι  $O((d + 1) \cdot n)$ .

5. To see an application of the k-NN algorithm in a real world classification problem consider the data found at <https://www.kaggle.com/uciml/iris>. Download from there the Iris.csv file. Ignore the id column and consider the columns: SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm as point coordinates in a four dimensional space. Consider the column Species as the class/label column. The file contains 150 rows. Run the algorithm on the first 100 rows and make predictions for the rest 50. How your predictions are compared to the actual? Explain your methodology.

### Thought Process

Στο ερώτημα αυτό εργαστήκαμε με βάση την υλοποίηση του αλγορίθμου k-NN που κατασκευάσαμε στο ερώτημα 1. Χρησιμοποιήσαμε τον κώδικα αυτόν με train dataset το Iris.csv, που περιέχεται στο παραδοταίο αρχείο και μπορεί κανείς να το αποκτήσει με τον τρόπο που περιγράφεται στην εκφώνηση της παρούσας ερώτησης.

Το σύνολο δεδομένων απαρτίζεται από αντικείμενα που αναπαριστούν ίριδες. Κάθε αντικείμενο διαθέτει έναν αναγνωριστικό αριθμό (id) και τα χαρακτηριστικά SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm, που κωδικοποιούν το μήκος και το πλάτος των φύλλων και των πετάλων σε εκατοστά (cm). Επίσης, τα αντικείμενα διαθέτουν ένα ακόμη χαρακτηριστικό που καλείται Species και είναι ένα string με το είδος του κάθε αντικειμένου. Το παρόν σύνολο δεδομένων περιέχει 3 είδη ίριδας: "Iris-setosa", "Iris-versicolor" και "Iris-virginica".



Figure 15: Iris Setosa



Figure 16: Iris Versicolor



Figure 17: Iris Virginica

To dataset που βρίσκεται στο Iris.csv αποτελείται από 150 εγγραφές. Ωστόσο, επειδή είναι "ταξινομημένο" σύμφωνα με το είδος, δηλαδή οι πρώτες 50 εγγραφές είναι εγγραφές που ανήκουν στην κατηγορία "Iris-setosa", οι επόμενες 50 στην κατηγορία "Iris-versicolor" και οι τελευταίες 50 στην κατηγορία "Iris-virginica", έπρεπε να "ανακατέψουμε" τα records του dataset και έπειτα να προχωρήσουμε στην κατηγοριοποίηση με k-NN. Η μεθοδολογία που χρησιμοποιήσαμε περιγράφεται αναλυτικότερα παρακάτω.

## Implementation

Όπως ήδη έχουμε αναφέρει χρησιμοποιήσαμε για την κατηγοριοποίηση των αντικειμένων τον k-NN που υλοποιήθηκε στο ερώτημα 1, χωρίς να προβούμε σε κάποια αλλαγή. Χρειάστηκε να δημιουργήσουμε μερικές ακόμα συναρτήσεις για την καταλληλη επεξεργασία του συγκεκριμένου συνόλου δεδομένων.

Αρχικά, υλοποιήσαμε την συνάρτηση `read_csv` η οποία είναι υπεύθυνη για το διάβασμα των δεδομένων από το αρχείο csv. Η ίδια συνάρτηση επεξεργάζεται τα χαρακτηριστικά των εγγραφών. Συγκεκριμένα, απομακρύνει το id, διότι είναι ένα χαρακτηριστικό που δεν περιέχει απολύτως καμία χρήσιμη πληροφορία για τον αλγόριθμο και αν επιλέγαμε να το διατηρήσουμε και ο αλγόριθμος να το λαμβάνει υπ' όψιν κατά τη κατηγοριοποίηση, τότε θα παρήγαγε λανθασμένα αποτελέσματα, για προφανείς λόγους. Επίσης, αντικαθιστούμε το χαρακτηριστικό "Species" με

έναν ακέραιο αριθμό στο διάστημα  $[0, 2]$ , αφού έχουμε μόνο 3 κατηγορίες. Η αντιστοίχιση γίνεται ως εξής:

- 0 → Iris-setosa
- 1 → Iris-versicolor
- 2 → Iris-virginica

με τη χρήση κατάλληλου dictionary. Τέλος, όλα τα πεδία μετατρέπονται σε float πεδία.

```
# read dataset from csv file          1
# give appropriate delimiter and labels of dataset    2
def read_from_csv(filename, delimiter, labels):      3
    dataset = []                                4
    with open(filename, 'r') as csv_file:        5
        csv_reader = reader(csv_file, delimiter=delimiter)  6
        header = next(csv_reader) # avoid header of csv file 7
        if header == None: # empty csv file           8
            return dataset
        else:                                         9
            for row in csv_reader:                  10
                del row[0] # remove id             11
                # change label from string to 0,1,2   12
                row.append(labels[row[-1]])         13
                del row[-2] # remove string label   14
                for idx in range(len(row)): # convert string data to float 15
                    data                         16
                    row[idx] = float(row[idx])       17
                dataset.append(row)               18
            return dataset                     19
```

Ακόμα, υλοποιήσαμε τη συνάρτηση `construct_folds`, η οποία χωρίζει το `dataset` σε `number_of_folds` ισάριθμα (από άποψη πλήθους εγγραφών) `folds`. Όπως ήδη αναφέραμε, το συγκεκριμένο `dataset` είναι "ταξινομημένο" ως προς το πεδίο "Species", γι' αυτό επιλέξαμε να "ανακατέψουμε" τα `records`. Ουσιαστικά, κάθε φορά που πρέπει να επιλεγεί ένα `record` από το `dataset` για να εισαχθεί στο τρέχον υπό κατασκευή `fold`, επιλέγεται τυχαία κάποιο και διαγράφεται, ώστε να μη μπορεί να επιλεγεί ξανά.

```
# dataset into k folds          1
def construct_folds(dataset, number_of_folds):    2
    records_per_fold = int(len(dataset) / number_of_folds)  3
    dataset_folds = []                                4
    for i in range(number_of_folds):                 5
        fold = [] # construct fold                 6
        for j in range(records_per_fold):           7
            # pick random record for fold          8
            random_index = random.randrange(len(dataset))  9
            record = dataset[random_index]           10
            fold.append(record)                      11
            del dataset[random_index]               12
        dataset_folds.append(fold)                   13
```

```
dataset_folds.append(fold)  
return dataset_folds
```

14  
15

---

Η πλήρης υλοποίηση βρίσκεται στο αρχείο exercise5.py του παραδοταίου.

## Running the code

```
$ python -m venv .env  
$ source .env/bin/activate  
$ pip install -r requirements.txt  
$ python exercise5.py
```

Figure 18: Οδηγίες εκτέλεσης προγράμματος

## References

- [1] Understanding Machine Learning: From Theory to Algorithms, Shai Shalev-Shwartz and Shai Ben-David, Cambridge University Press, 2014, pages. 258-267, ISBN: 978-1-107-05713-5
- [2] Πολύτροπη Μείωση Διάστασης Δεδομένων με Χρήση Πυρήνα, Διπλωματική εργασία, Πέτρος Χ. Δραχούλης, Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, Τμήμα Πληροφορικής, 2016
- [3] Pattern recognition and machine learning, Christopher Bishop, Springer, 2006, pages. 33-38, ISBN: 978-0-387-31073-2
- [4] Εισαγωγή στην εξόρυξη δεδομένων, Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Εκδόσεις Τζιόλα, 2017, pages. 57-59, 78-80, 247-251, ISBN: 978-960-418-162-9