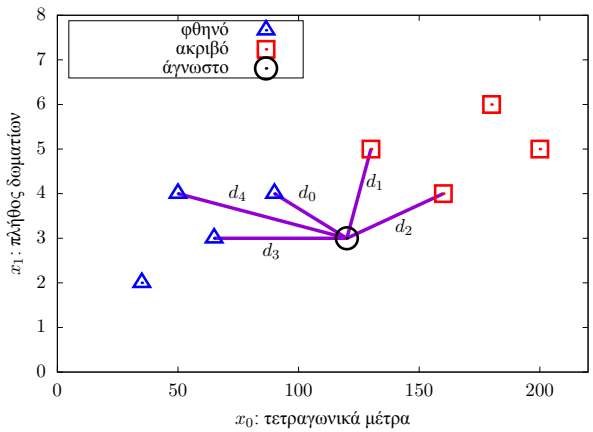
Πανεπιστήμιο Πειραιώς
Σχολή Τεχνολογιών Πληροφορικής και Επικοινωνιών
Τμήμα Ψηφιακών Συστημάτων
Δομές Δεδομένων 2017-2018 – 1η Εργασία

Χρήστος Δουλκερίδης Ορέστης Τελέλης

# 1 Περιγραφή

Η μέθοδος των *«$k$-Εγγύτερων Γειτόνων»* (*$k$-Nearest Neighbors*: $k$−NN) χρησιμοποιείται στη Μηχανική Μάθηση για αυτόματη κατηγοριοποίηση. Εκτιμά την άγνωστη κατηγορία $y$ (από πεπερασμένο σύνολο κατηγοριών) που αντιστοιχεί σε μετρήσεις $n$ μεγεθών, οι οποίες αναπαρίστανται ως $n$-διάστατο σημείο $x = (x_0, \ldots, x_{n-1})$. Για την εκτίμηση, η μέθοδος αξιοποιεί τις *δεδομένες* κατηγορίες $y^{(i)}$ για $m$ *δεδομένα* $n$-διάστατα σημεία μετρήσεων $x^{(i)}$, $i = 0, \ldots, m-1$. Στην εργασία θα υλοποιήσετε δομή απλά συνδεδεμένης λίστας που υποστηρίζει λειτουργίες της μεθόδου $k$−NN και θα πειραματιστείτε με αυτήν.

**Η Μέθοδος $k$−NN με Παράδειγμα.** Στο παρακάτω σχήμα απεικονίζονται 8 σημεία $x^{(i)} = (x_0^{(i)}, x_1^{(i)})$, $i = 0, 1, \ldots, 7$, όπου το $x_0$ μετρά την επιφάνεια ενός σπιτιού σε τετραγωνικά μέτρα και το $x_1$ μετρά το πλήθος των δωματίων. Τα σπίτια που σημαδεύονται με «τρίγωνο» θεωρούνται φθηνά, ενώ αυτά που σημαδεύονται με τετράγωνο θεωρούνται ακριβά.




***Υπάρχοντα Δεδομένα***

| *τ.μ.* $x_0^{(i)}$ | *Δωμάτια* $x_1^{(i)}$ | *Κατηγορία* $y^{(i)}$ |
|---|---|---|
| 35 | 2 | φθηνό |
| 50 | 4 | φθηνό |
| 65 | 3 | φθηνό |
| 90 | 4 | φθηνό |
| 130 | 5 | ακριβό |
| 160 | 4 | ακριβό |
| 180 | 6 | ακριβό |
| 200 | 5 | ακριβό |

Από την περιγραφή $x = (x_0, x_1)$ ενός σπιτιού που είδαμε σε αγγελία, θέλουμε να εκτιμήσουμε αν κατηγοριοποιείται σε αυτά που θεωρούμε φθηνά, βάσει των δεδομένων μας, ή αν είναι ακριβό. Μία εκδοχή της μεθόδου $k$−NN εφαρμόζεται ως εξής: για δεδομένη τιμή του $k$, έστω $k = 5$, εντοπίζουμε τα $k$ εγγύτερα σημεία $x^{(i)}$ στο $x$, ως προς τις Ευκλείδειες αποστάσεις τους. Έστω $d_0$, $d_1$, $d_2$, $d_3$, $d_4$ οι αποστάσεις των $k$ εγγύτερων σημείων (δείτε το σχήμα). Μετράμε τη «βαρύτητα» κάθε κατηγορίας («φθηνό/ακριβό»), βάσει των αποστάσεων, ως εξής. Η βαρύτητα της κατηγορίας «φθηνό» υπολογίζεται ως το άθροισμα $\frac{1}{d_0} + \frac{1}{d_3} + \frac{1}{d_4}$, που προκύπτει από τις αποστάσεις των εγγύτερων γειτόνων που κατηγοριοποιούνται ως «φθηνοί». Αντίστοιχα, η βαρύτητα της κατηγορίας «ακριβό» υπολογίζεται ως το άθροισμα $\frac{1}{d_1} + \frac{1}{d_2}$. Με τον τρόπο αυτόν, καθένας από τους $k = 5$ εγγύτερους γείτονες συνεισφέρει «ψήφο» για το αγνώστου κατηγορίας σημείο, αντιστρόφως ανάλογη προς την απόστασή του από αυτό. Η κατηγορία του σημείου $x$ ορίζεται ως εκείνη με τη μεγαλύτερη βαρύτητα.

***Ευκλείδεια Απόσταση.*** Η Ευκλείδεια απόσταση $d(x, x')$, μεταξύ δύο $n$-διάστατων σημείων $x$ και $x'$ είναι:

$$d(x, x') = \sqrt{(x_0 - x'_0)^2 + (x_1 - x'_1)^2 + \cdots + (x_{n-1} - x'_{n-1})^2}$$

# 2 Ζητούμενα Εργασίας

## 2.1 Υλοποίηση

Θα υλοποιήσετε τις κλάσεις `PointData`, `Node`, `PointList` και `KNN`, που προδιαγράφονται στο *δεδομένο* αρχείο `KNN.java`. Στην υλοποίηση των μεθόδων ***δε θα πρέπει να χρησιμοποιήσετε arrays (πίνακες)***, παρά μόνο για αναπαράσταση σημείων $x$, όπου χρειάζεται. Θα αξιοποιήσετε κατάλληλα τις μεθόδους της λίστας `PointList` που θα υλοποιήσετε. ***Μπορείτε να προσθέσετε μόνο `private` πεδία ή μεθόδους***.

### 2.1.1 Κλάσεις `PointData` και `Node`

Ένα αντικείμενο της κλάσης `PointData` αποθηκεύει απλώς ένα σημείο $x^{(i)}$, μαζί με την κατηγοριοποίησή του $y^{(i)}$, και προσφέρει μία μέθοδο Ευκλείδειας απόστασης `dist` του σημείου $x^{(i)}$ από άλλο δεδομένο σημείο $x$. Για αποφυγή σφαλμάτων, ο κατασκευαστής της `PointData` θα πρέπει ***να αντιγράφει*** το σημείο εισόδου `x` στο πεδίο `this.x` και όχι να αναθέτει την αναφορά με `this.x = x`.

```
class PointData {
  public double[] x;
  public double y;

  public PointData(double[] x, double y)
  public double dist(double[] x)
}
```

```
class Node {
  public double score;
  public PointData ptd;
  public Node next;

  public Node(double[] x, double y)
}
```

Ο κατασκευαστής της κλάσης `Node` δημιουργεί κόμβο λίστας από δεδομένο σημείο (και την κατηγορία του), αρχικοποιώντας το πεδίο `ptd`. Η κλάση `Node` περιέχει ένα πεδίο `score` τύπου `double`, που μπορεί να χρησιμοποιηθεί στην υλοποίηση της μεθόδου `classify` (περιγράφεται παρακάτω).

### 2.1.2 Κλάση `PointList`

Υλοποιεί δομή απλά συνδεδεμένης λίστας που παρέχει τις ακόλουθες μεθόδους.

`public PointList(int dim)` Κατασκευάζει κενή λίστα κλάσης `PointList` που αποθηκεύει ζεύγη $(x, y)$, όπου το $x$ είναι διάστασης `dim`.

`public static PointList readData(String filename)` Κατασκευάζει και επιστρέφει μία νέα λίστα κλάσης `PointList`, διαβάζοντας ένα σύνολο δεδομένων από αρχείο με όνομα `filename` και εισάγοντας τα δεδομένα ζεύγη στη λίστα. Αυτή η μέθοδος δίνεται υλοποιημένη για τη διευκόλυνσή σας. Χρησιμοποιείται ως εξής: `PointList list = PointList.readData("wine.dat");`

`public int getDim()` Επιστρέφει τη διάσταση των σημείων που αποθηκεύονται στη λίστα.

`public int length()` Επιστρέφει το πλήθος των κόμβων της λίστας.

`public boolean append(double[] x, double y)` Εισάγει αντίγραφο του ζεύγους εισόδου `x,y`, ενθυλακωμένο σε αντικείμενο της κλάσης `PointData`, στο τέλος της λίστας και επιστρέφει `true`. Επιστρέφει `false` αν το `x` είναι διαφορετικού μήκους από τη διάσταση των σημείων που αποθηκεύει η λίστα.

`public boolean append(PointList list)` Επισυνάπτει τα περιεχόμενα της λίστας εισόδου `list`, στο τέλος της λίστας στην οποία καλείται η μέθοδος, *αντιγράφοντας κάθε σημείο* κλάσης `PointData` που περιέχει η `list`, δίχως να διασυνδέει τη list με τη λίστα στην οποία καλείται η μέθοδος[1]. Επιστρέφει `false`, αν οι δύο λίστες αποθηκεύουν σημεία διαφορετικών διαστάσεων. Αλλιώς, επιστρέφει `true`.

`public PointData rmFirst()` Διαγράφει τον πρώτο κόμβο της λίστας και επιστρέφει αναφορά στο αντικείμενο κλάσης `PointData` που ενθυλακώνει τα δεδομένα. Αν η λίστα είναι κενή επιστρέφει `null`.

**Σημείωση:** ***Όλες οι ακόλουθες μέθοδοι μπορούν να μεταβάλλουν τη σχετική σειρά με την οποία αποθηκεύονται τα δεδομένα της λίστας.***

[1] Υπόδειξη Java: οι μέθοδοι μίας κλάσης έχουν πρόσβαση στα `private` πεδία όλων των αντικειμένων της κλάσης.

`public void shuffle()` *«Ανακατεύει»* τυχαία και ομοιόμορφα τη σειρά με την οποία αποθηκεύονται τα δεδομένα κλάσης `PointData` της λίστας.

`public PointData rmNearest(double[] x)` Εντοπίζει και διαγράφει από τη λίστα τον κόμβο που αποθηκεύει τα δεδομένα του εγγύτερου σημείου στο σημείο εισόδου `x`. Επιστρέφει αναφορά κλάσης `PointData` στα δεδομένα αυτά, εκτός αν η λίστα είναι κενή, οπότε επιστρέφει `null`.

`public PointList findKNearest(double[] x, int k)` Για τη δεδομένη τιμή του `k`, εντοπίζει στη λίστα τους `k` εγγύτερους γείτονες του σημείου εισόδου `x`. Εγγράφει σε νέα λίστα κλάσης `PointList` αυτούς τους $k$ εγγύτερους γείτονες, την οποία επιστρέφει.

`public double classify(double[] x)` Κατηγοριοποιεί το σημείο εισόδου `x`, βάσει των σημείων που αποθηκεύονται στη λίστα στην οποία καλείται η μέθοδος (θεωρώντας ότι *όλα* τα σημεία που περιέχει η λίστα είναι «εγγύτεροι γείτονες» του `x`). Η μέθοδος θα πρέπει να δουλεύει ορθά για *οσεσδήποτε* διαφορετικές κατηγορίες εμφανίζονται στα περιεχόμενα της λίστας. Για την υλοποίησή της, θα σας φανεί χρήσιμο το `public` πεδίο `score`, που περιέχεται στην κλάση `Node`. Όπως και οι υπόλοιπες μέθοδοι, αναμένεται και αυτή να υλοποιηθεί δίχως τη χρήση arrays (πινάκων). Εφόσον έχετε υλοποιήσει σωστά τις μεθόδους `classify` και `findKNearest`, ο αλγόριθμος $k$−NN για ένα σημείο `x` και δεδομένη τιμή του $k$ (π.χ., $k = 5$) εκφράζεται σε μία γραμμή: `list.findKNearest(x, 5).classify(x);`

### 2.1.3 Κλάση KNN

Αυτή θα είναι η κύρια κλάση του προγράμματός σας, που υλοποιεί τη μέθοδο `main`. Η μέθοδος `main` θα εκτελεί τον πειραματισμό που περιγράφεται στην ακόλουθη παράγραφο.

## 2.2 Πειραματισμός και Τεχνική Αναφορά

Δίνονται δύο (2) αρχεία δεδομένων. Για το κάθε αρχείο και για κάθε τιμή του $k \in \{1, 2, \ldots, 10\}$, θα επαναλάβετε 10 φορές την εξής διαδικασία. Θα δημιουργήσετε δύο λίστες της κλάσης `PointList`. Η μία λίστα («μικρή») θα περιλαμβάνει ένα τυχαία και ομοιόμορφα επιλεγμένο 25% των δεδομένων του αρχείου και η άλλη το υπόλοιπο 75% («μεγάλη»). Στη συνέχεια, θα μετρήσετε την ακρίβεια κατηγοριοποίησης του αλγορίθμου $k$−NN για την τρέχουσα τιμή του $k$, ως το ποσοστό των σημείων της «μικρής» λίστας που κατηγοριοποιούνται ορθά από τη μέθοδο. Πιο αναλυτικά, θα κατηγοριοποιήσετε με τον αλγόριθμο κάθε σημείο της «μικρής» λίστας, χρησιμοποιώντας τα σημεία της «μεγάλης» λίστας, και θα μετρήσετε το ποσοστό των σημείων της «μικρής» λίστας που κατηγοριοποιήθηκαν ορθά. Το πρόγραμμά σας θα εκτυπώνει τη *μέση ακρίβεια* κατηγοριοποίησης για το τρέχον αρχείο δεδομένων και την τρέχουσα τιμή του $k$, υπολογισμένη πάνω από τις 10 επαναλήψεις.

Θα συντάξετε μία τεχνική αναφορά, στην οποία θα δείξετε και θα σχολιάσετε τα αποτελέσματα του πειραματισμού σας. Στην αναφορά, θα περιλαμβάνονται πίνακες, ένας για κάθε αρχείο δεδομένων, όπου θα καταγράφεται η μέση ακρίβεια κατηγοριοποίησης του αλγορίθμου, για κάθε διαφορετική τιμή του $k$. Επιπλέον θα παράξετε ραβδογράμματα που απεικονίζουν τα αποτελέσματα των πινάκων (ένα για κάθε αρχείο δεδομένων).

Η τεχνική αναφορά θα περιλαμβάνει συνοπτική περιγραφή υλοποίησης, για καθεμία από τις μεθόδους `shuffle`, `rmNearest`, `findKNearest`, `classify`, της κλάσης `PointList` – με ψευδοκώδικα ή/και επεξηγήσεις όπου χρειάζεται. Με βάση την περιγραφή σας, θα αναλύσετε την πολυπλοκότητα της υλοποίησής σας για καθεμία από αυτές τις μεθόδους, ως προς τις παραμέτρους: *μήκος λίστας* $m$ *(ή* $k$*, όπου χρειάζεται)*, *διάσταση σημείων* $n$. Συμπερασματικά, θα εξάγετε και θα αιτιολογήσετε την πολυπλοκότητα της υλοποίησης του αλγορίθμου $k$−NN.

**Αρχεία Δεδομένων.** Είναι απλά αρχεία κειμένου. Στην 1η γραμμή περιέχουν το πλήθος $m$ των γραμμών δεδομένων και τη διάσταση $n$ των σημείων $x^{(i)}$. Ακολουθεί μία γραμμή για κάθε ζεύγος $(x^{(i)}, y^{(i)})$, που περιέχει $n+1$ πραγματικούς αριθμούς, χωριζόμενους με ένα κενό χαρακτήρα: Οι $n$ πρώτοι αριθμοί αντιστοιχούν στις συντεταγμένες του $x^{(i)}$. Ο τελευταίος αριθμός αντιστοιχεί στην κατηγορία $y^{(i)}$. Το σύνολο δεδομένων στο αρχείο `wine.dat` αφορά στην κατηγοριοποίηση κρασιού σε τρεις κλάσεις. Το σύνολο δεδομένων στο αρχείο `housing.dat` αφορά στην κατηγοριοποίηση σπιτιών σε «φθηνό/ακριβό».

## 3 Διαδικαστικά Θέματα

**ΠΑΡΑΔΟΤΕΑ** Θα πρέπει να παραδώσετε ένα αρχείο ΑΜ_Επώνυμο_Όνομα.zip (όπου ΑΜ ο αριθμός μητρώου) που θα περιλαμβάνει **δύο (2) αρχεία μόνο:**

1. Το αρχείο πηγαίου κώδικα **KNN.java**, επαρκώς σχολιασμένο.
   **Προσοχή:** σχόλια στα ελληνικά, μόνο σε κωδικοποίηση UTF-8. Άλλες κωδικοποιήσεις (ISO-8859-7, Windows-1253) *απορρίπτονται από τον Java Compiler*. Eclipse/Netbeans: δεξί click στο Project, επιλογή Properties και στην ιδιότητα «Encoding» επιλέγετε UTF-8.
2. Την τεχνική αναφορά σας **σε μορφή pdf**. **ΘΑ ΑΓΝΟΗΘΕΙ** *κάθε άλλη μορφή εγγράφου*.

**ΠΑΡΑΔΟΣΗ αποκλειστικά μέσω της πλατφόρμας «ΕΥΔΟΞΟΣ»** του τμήματος, έως και την **15/12/2017, 23:59**. Ανεβάστε το ΑΜ_Επώνυμο_Όνομα.zip στην περιοχή «Εργασίες».

**ΕΡΩΤΗΣΕΙΣ/ΑΠΟΡΙΕΣ/ΔΙΕΥΚΡΙΝΙΣΕΙΣ** αποκλειστικά μέσα από την Περιοχή Συζήτησης «ΕΡΓΑΣΙΑ 1» της πλατφόρμας «ΕΥΔΟΞΟΣ». Δε θα απαντηθούν Email με απορίες.

**ΔΕ ΘΑ ΑΞΙΟΛΟΓΗΘΟΥΝ:**

- Εργασίες που θα παραδοθούν εκπρόθεσμα ή με άλλο τρόπο (email, CD κ.λ.π.)
- Εργασίες που παραδίδονται σε μορφή συμπίεσης διαφορετική από .zip.
- Εργασίες που περιλαμβάνουν μέσα άσχετα αρχεία.

**ΒΑΘΜΟΛΟΓΗΣΗ** Η εργασία μετρά 20% στον τελικό βαθμό (εφόσον το γραπτό σας βαθμολογηθεί με τουλάχιστον 4).

- Φτωχή / Δυσανάγνωστη τεκμηρίωση ή έλλειψη αυτής στην αναφορά για κάποια λειτουργία, επιφέρει άμεση απώλεια 50% του μέγιστου δυνατού βαθμού για τη λειτουργία αυτή.
- Η τεχνική αναφορά ή τμήματα αυτής δε θα ληφθούν υπόψην στη βαθμολόγηση αν δε συνοδεύονται από κώδικα που μεταγλωττίζεται και εκτελείται.
- **Η εργασία είναι αυστηρά ατομική.** Αντιγραφή στον κώδικα και/ή στην τεχνική αναφορά επιφέρει άμεσο μηδενισμό της εργασίας. Οι εργασίες θα ελεγχθούν (όλες ανά ζεύγη).