# **Εικόνα που περιέχει έμβλημα, κέρμα, χρυσό Περιγραφή που δημιουργήθηκε αυτόματαΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ**

Περιεχόμενα

[**ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ** 1](#_Toc136783815)

[**Πρόβλημα 3.1** 3](#_Toc136783816)

[**ΑΠΟΤΕΛΕΣΜΑΤΑ** 3](#_Toc136783817)

[**Πρόβλημα 3.2** 4](#_Toc136783818)

[**α)** 4](#_Toc136783819)

[**β)** 4](#_Toc136783820)

[**γ)** 5](#_Toc136783821)

[**δ)** 7](#_Toc136783822)

[**ε)** 7](#_Toc136783823)

[**ΑΠΟΤΕΛΕΣΜΑΤΑ** 8](#_Toc136783824)

[**Πρόβλημα 3.3** 9](#_Toc136783825)

[**α)** 9](#_Toc136783826)

[**ΑΠΟΤΕΛΕΣΜΑΤΑ** 9](#_Toc136783827)

[**β)** 10](#_Toc136783828)

[**ΑΠΟΤΕΛΕΣΜΑΤΑ** 10](#_Toc136783829)

[**Κώδικας Προβλήματος 3.1** 11](#_Toc136783830)

[**Κώδικας Προβλήματος 3.2** 12](#_Toc136783831)

[**Κώδικας Προβλήματος 3.3** 13](#_Toc136783832)

## **Πρόβλημα 3.1**

Στο πρόβλημα 3.1 θα χρησιμοποιήσουμε την μέθοδο του Γκαουσιανού πυρήνα για την προσέγγιση της συνάρτησης πυκνότητας πιθανότητας ενός συγκεκριμένου συνόλου δεδομένων . Η θεμελιώδης ιδέα πίσω από αυτή την τεχνική είναι η εκτίμηση πυκνότητας πυρήνα (KDE), η οποία είναι ένας τρόπος εκτίμησης της συνάρτησης πυκνότητας πιθανότητας μιας τυχαίας μεταβλητής. Οι μέθοδοι πυρήνα είναι μη παραμετρικές, πράγμα που σημαίνει ότι δεν απαιτούν προηγούμενες υποθέσεις σχετικά με την κατανομή της τυχαίας μεταβλητής.

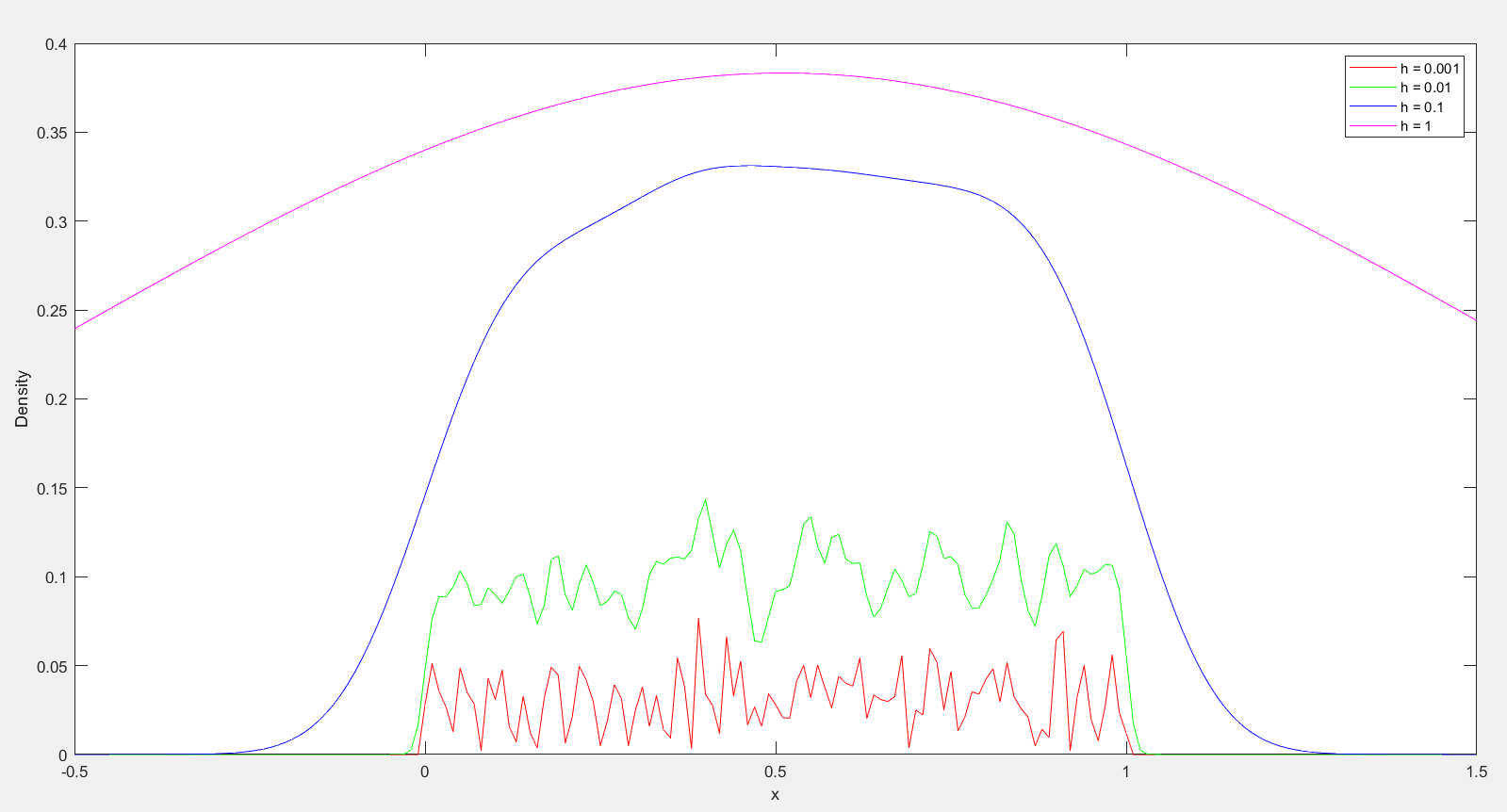
Η διαδικασία που ακολουθήθηκε είναι η εξής:

🡪Αρχικοποιούμε N = 1000 τον αριθμό των τυχαίων μεταβλητών που θέλουμε να δημιουργήσουμε και δημιουργούμε ένα τυχαίο δείγμα Ν σημείων ομοιόμορφα κατανεμημένων στο [0, 1].

🡪 Ορίζουμε το εύρος ζώνης το οποίο είναι μια παράμετρος της μεθόδου που επηρεάζει την ομαλότητα της συνάρτησης πυκνότητας που προκύπτει. Μικρότερες τιμές εύρους ζώνης αποδίδουν πιο λεπτομερείς εκτιμήσεις πυκνότητας, ενώ μεγαλύτερες τιμές εύρους ζώνης αποδίδουν πιο ομαλές εκτιμήσεις.

🡪 Δημιουργούμε ένα εύρος τιμών x για το οποίο θα εκτιμηθεί το PDF. Το PDF δεν περιορίζεται στο διάστημα [0, 1]. Για κάθε τιμή x και κάθε σημείο δεδομένων, υπολογίζεται ο πυρήνας Gauss. Στη συνέχεια, αυτές οι τιμές του πυρήνα υπολογίζονται κατά μέσο όρο για να προκύψει η τιμή KDE στη συγκεκριμένη τιμή x. Η διαδικασία επαναλαμβάνεται για όλες τις τιμές x για να ληφθεί το πλήρες KDE.

### **ΑΠΟΤΕΛΕΣΜΑΤΑ**



Η μέθοδος πυρήνα λειτουργεί με βάση την αρχή της εξομάλυνσης των δεδομένων. Ειδικότερα, ο πυρήνας Gauss παίρνει κάθε σημείο δεδομένων στο σύνολο και το αντικαθιστά με μια κατανομή Gauss με κέντρο το συγκεκριμένο σημείο δεδομένων. Το "πλάτος" αυτής της κατανομής καθορίζεται από το εύρος ζώνης "h". Η τελική εκτίμηση της πυκνότητας σε κάθε σημείο είναι ο μέσος όρος των συναρτήσεων πυρήνα με κέντρο κάθε σημείο δεδομένων. Με άλλα λόγια, είναι ο μέσος όρος των Γκαουσιανών κατανομών που αντικαθιστούν κάθε σημείο δεδομένων.

Η επιλογή του εύρους ζώνης h είναι καθοριστικής σημασίας. Αν το h είναι πολύ μικρό, η εκτίμηση είναι πολύ τραχιά, ενώ αν το h είναι πολύ μεγάλο, η εκτίμηση είναι πολύ ομαλή και μπορεί να χάσει σημαντικά χαρακτηριστικά. Αυτό είναι γνωστό ως συμβιβασμός μεροληψίας-διακύμανσης στη στατιστική. Το ιδανικό h ούτε υπεραπλουστεύει ούτε υπεραπλοποιεί τα δεδομένα.

## **Πρόβλημα 3.2**

### **α)**

Το Representer theorem δηλώνει ότι η συνάρτηση που ελαχιστοποιεί τη κανονικοποιημένη εμπειρική συνάρτηση κινδύνου βρίσκεται στο εύρος του πυρήνα που αξιολογείται στα δείγματα εκπαίδευσης. Είναι ο ακρογωνιαίος λίθος των μεθόδων πυρήνα στη μηχανική μάθηση, καθώς μας επιτρέπει να δουλεύουμε μόνο με έναν πεπερασμένο αριθμό συντελεστών, αντί για έναν άπειρο-διάστατο συναρτησιακό χώρο.

*Βασική ιδιότητα*

Φ(Χ)= ανήκει στον V. Αν Κ(Χ,Χ0) στο V τότε

<Κ(Χ,Χ0), Φ(Χ)> = Φ(Χ0) = = Φ(Χ0)

Η συνάρτηση ϕ(X) που θέλουμε να βελτιστοποιήσουμε μπορεί να ξαναγραφεί ως Φ(X) = η οποία, σύμφωνα με τη Βασική Ιδιότητα, ανήκει στον V.

Το θεώρημα μας λέει ότι η βέλτιστη συνάρτηση ϕ μπορεί να γραφτεί ως γραμμικός συνδυασμός συναρτήσεων πυρήνα με κέντρο τα σημεία δεδομένων. Γι' αυτό μπορούμε να αντικαταστήσουμε το ϕ(X) με το ϕ\_hat(X), όπου το ϕ\_hat(X) ορίζεται ως η ορθογώνια προβολή του ϕ στον υπόχωρο που καλύπτεται από τους πυρήνες που είναι κεντραρισμένοι γύρω από τα σημεία δεδομένων.

Τυπικά, για κάθε X:

Φ\_hat(X)= *Σ*Xi ϵ stars [ai\* K(X,Xi)] + *Σ*Xj ϵ circles [bj\* K(X,Xj)]

### **β)**

Η αρχή της ορθογωνιότητας μας λέει ότι η διαφορά μεταξύ μιας συνάρτησης ϕ(X) και της ορθογώνιας προβολής της ϕ\_hat(X) σε έναν υπόχωρο Ω είναι ορθογώνια σε κάθε διάνυσμα στον Ω.

Εφαρμόζοντας αυτή την αρχή, γνωρίζουμε ότι για κάθε Xi:

<Φ(X)-Φ\_hat, K(X,Xi)> = 0

Αυτό απλοποιείται ως εξής:

<Φ(X), K(X,Xi)> = <Φ\_hat, K(X,Xi)>

Ως αποτέλεσμα, έχουμε ότι Φ(X) = Φ\_hat. Επομένως, αναπτύσσοντας την νόρμα:

||Φ(X)||2 = || Φ\_hat(X) + Φ(X) - Φ\_hat(X)||2

= ||Φ(X)||2 + |||Φ(X) - Φ\_hat(X)||2 + 2\*< Φ\_hat(X), Φ(X) - Φ\_hat(X)>

Ο τελευταίος όρος είναι μηδενικός επειδή η Φ\_hat(X) είναι ορθογώνια προς τη Φ(X) - Φ\_hat(X). Έτσι, μας μένει το εξής:

||Φ(X)||2 = ||Φ\_hat(X)||2 + ||Φ(X) - Φ\_hat(X)||2

Επειδή όλες οι νόρμες είναι μη αρνητικές, μπορούμε να πούμε:

||Φ(X)||2 >= ||Φ\_hat(X)||2

Αυτό δείχνει ότι μπορούμε να αντικαταστήσουμε τον εξομαλυντή ||ϕ(X)||2 στην αρχική συνάρτηση κόστους με ||ϕ\_hat(X)||2. Αυτή η αντικατάσταση απλοποιεί το πρόβλημα βελτιστοποίησης ώστε να περιλαμβάνει μόνο τους πεπερασμένους συντελεστές ai και bj, οι οποίοι είναι πολύ πιο εύκολο να υπολογιστούν.

### **γ)**

Η διαδικασία επίλυσης περιλαμβάνει τη διαφόριση της κανονικοποιημένης εμπειρικής συνάρτησης κινδύνου και τον μηδενισμό της παραγώγου για την επίλυση των ai και bj.

Ωστόσο, πριν προχωρήσουμε σε αυτό, ας τροποποιήσουμε τη συνάρτηση κόστους μας ώστε να συμπεριλάβουμε τη λύση του θεωρήματος Representer (ϕ\_hat αντί για ϕ):

Ξεκινάμε με:

min { *Σ*Xi ϵ stars [1-ϕ(Xi)2] + *Σ*Xj ϵ circles [1+ϕ(Xj)]2 + λ\*|||ϕ(X)||2 }

Και αντικαθιστούμε το ϕ με το ϕ\_hat όπως προκύπτει από το Representer θεώρημα:

min { *Σ*Xi ϵ stars 1-(sum ai \* K(X, Xi))2] + *Σ*Xj ϵ circles [1+( *Σ* bj \* K(X, Xj))2] + λ\* ||ϕ\_hat(X)||2 }

Εδώ, ελαχιστοποιούμε πάνω στο ai (για όλα τα Xi σε αστέρια) και στο bj (για όλα τα Xj σε κύκλους).

Σημειώνουμε ότι ο όρος λ\*||ϕ\_hat(X)||2 μπορεί επίσης να εκφραστεί με όρους των ai, bj και της συνάρτησης πυρήνα. Σύμφωνα με τον ορισμό του ϕ\_hat και τις ιδιότητες της συνάρτησης πυρήνα, έχουμε:

||ϕ\_hat(X)||2 = <ϕ\_hat(X), ϕ\_hat(X)>

= < *Σ*Xi ϵ stars ai \* K(X, Xi)] + *Σ*Xj ϵ circles bj \* K(X, Xj)] , *Σ*Xi ϵ stars ai \* K(X, Xi)] + *Σ*Xj ϵ circles [bj \* K(X, Xj)]>

= *Σ*Xi,Xi’ ϵ stars [ai \* ai' \* K(Xi, Xi')] + 2 \* *Σ*Xi ϵ stars, Xj ϵ circles [ai \* bj \* K(Xi, Xj)] + *Σ*Xj , Xj’ ϵ circles [bj \* bj' \*

K(Xj, Xj')].

όπου <., .> συμβολίζει το εσωτερικό γινόμενο στο χώρο των συναρτήσεων.

Θα πρέπει να διαφορίσουμε αυτή τη συνάρτηση κόστους ως προς κάθε ai και bj, να θέσουμε κάθε παράγωγο στο μηδέν και να λύσουμε το προκύπτον σύστημα εξισώσεων. Ο αριθμός των εξισώσεων στο σύστημα θα ήταν ίσος με τον συνολικό αριθμό των αστεριών και των κύκλων.

Στην πράξη, αυτό θα γινόταν αριθμητικά και μπορεί να είναι υπολογιστικά εντατικό για μεγάλα σύνολα δεδομένων. Σε αυτό το σημείο έρχονται σε εφαρμογή αλγόριθμοι μάθησης όπως ο SVM με ένα τέχνασμα πυρήνα, καθώς επιλύουν αποτελεσματικά αυτού του είδους προβλήματος βελτιστοποίησης.

Υλοποιήσαμε κώδικα σε Matlab που ακολουθεί την εξής διαδικασία:

🡪Ορίζουμε τη συνάρτηση Gaussian kernel που θα χρησιμοποιήσουμε στο πρόβλημά μας.

🡪 Συμπληρώνουμε έναν πίνακα πυρήνα, K, όπου κάθε είσοδος Kij είναι το αποτέλεσμα της εφαρμογής του γκαουσιανού πυρήνα στα σημεία δεδομένων i και j.

🡪 alpha = (K + lambda \* eye(size(K))) \ labels : Εδώ γίνεται η βελτιστοποίηση. Αυτή η γραμμή υπολογίζει τις τιμές του alpha (οι συντελεστές ai και bj) που ελαχιστοποιούν τη ρυθμισμένη συνάρτηση κινδύνου. Αυτό γίνεται με την επίλυση του συστήματος γραμμικών εξισώσεων (K + lambda \* eye(size(K))) \* alpha = ετικέτες για alpha. Εδώ, το eye(size(K)) είναι ένας ταυτοτικός πίνακας ίδιου μεγέθους με το K, οι ετικέτες είναι οι εκχωρημένες αριθμητικές ετικέτες (+1 για αστέρια, -1 για κύκλους) και \ είναι ο τελεστής του MATLAB για την αριστερή διαίρεση πινάκων, ο οποίος επιλύει αποτελεσματικά το σύστημα γραμμικών εξισώσεων

🡪 Έπειτα υπολογίζουμε την τιμή της συνάρτησης Φ\_hat σε κάθε σημείο ενός πλέγματος που καλύπτει το εύρος των δεδομένων. Αυτό χρησιμοποιείται για τη δημιουργία ενός περιγράμματος του ορίου απόφασης.

🡪 Ακολουθεί η δημιουργία του contour plot που υπολογίζει το Φ\_hat για ένα νέο τυχαίο σημείο και το κατηγοριοποιεί ως αστέρι ή κύκλο με βάση το αν το Φ\_hat είναι θετικό ή αρνητικό.

### **δ)**

Μόλις έχουμε το βέλτιστο ϕ\_hat(X), το χρησιμοποιούμε για να κατηγοριοποιήσουμε ένα νέο σημείο Xnew σε "αστέρι" ή "κύκλο". Για ένα δεδομένο Xnew, υπολογίζουμε την τιμή του κάτω από το ϕ\_hat(Xnew) ως εξής:

ϕ\_hat(Xnew) =

όπου ai είναι οι συντελεστές (ή βάρη) που καθορίζονται από τον αλγόριθμο μάθησης και K(Xnew,Xi) είναι η συνάρτηση πυρήνα που εφαρμόζεται μεταξύ του Xnew και κάθε σημείου των δεδομένων εκπαίδευσης.

Το ϕ\_hat(Xnew) δεν θα παίρνει γενικά τις ακριβείς τιμές 1 ή -1, καθώς το όριο μεταξύ "αστέρα" και "κύκλου" μπορεί να μην διαχωρίζει τέλεια τα δεδομένα. Επομένως, λαμβάνουμε μια απόφαση με βάση το πρόσημο του ϕ\_hat(Xnew). Εάν ϕ\_hat(Xnew) > 0, ταξινομούμε το Xnew ως "αστέρι" και εάν ϕ\_hat(Xnew) < 0, ταξινομούμε το Xnew ως "κύκλο".

### **ε)**

Για να απεικονίσουμε το διαχωριστικό όριο μεταξύ των κλάσεων σε έναν δισδιάστατο χώρο, υπολογίζουμε το ϕ\_hat(X) για ένα πλέγμα σημείων στο δισδιάστατο επίπεδο. Αν η τιμή είναι μεγαλύτερη από 0, είναι πιο κοντά στην κατηγορία "αστέρι", ενώ αν είναι μικρότερη από 0, είναι πιο κοντά στην κατηγορία "κύκλος". Το όριο είναι εκεί όπου το ϕ\_hat(X) ισούται με 0, όπου η απόφαση είναι διφορούμενη μεταξύ των δύο κατηγοριών.

Στον κώδικά μας δημιουργούμε ένα πλέγμα πάνω από το εύρος των δεδομένων μας, υπολογίζοντας το ϕ\_hat(X) για κάθε σημείο σε αυτό το πλέγμα, και στη συνέχεια χρησιμοποιώντας contour plot για να σχεδιάσει το όριο όπου το ϕ\_hat(X) είναι 0. Τα διαγράμματα διασποράς δείχνουν τα αρχικά σημεία των δεδομένων, και το κόκκινο σημείο αντιπροσωπεύει το νέο σημείο που κατηγοριοποιείται με βάση το ϕ\_hat(Xnew).

Το h είναι το εύρος ζώνης του γκαουσιανού πυρήνα, ελέγχει την "ομαλότητα" του ορίου απόφασης. Ένα μικρότερο h θα καθιστούσε το όριο πιο ευέλικτο (ενδεχομένως υπερβολική προσαρμογή στα δεδομένα εκπαίδευσης), ενώ ένα μεγαλύτερο h θα καθιστούσε το όριο πιο ομαλό (ενδεχομένως υποπροσαρμογή στα δεδομένα).

Το λ είναι η παράμετρος κανονικοποίησης που βοηθά στον έλεγχο της υπερπροσαρμογής. Μεγαλύτερες τιμές του λ αυξάνουν την κανονικοποίηση, οδηγώντας σε ένα απλούστερο μοντέλο (υποπροσαρμογή), ενώ μικρότερες τιμές του λ μειώνουν την κανονικοποίηση, επιτρέποντας μεγαλύτερη πολυπλοκότητα στο μοντέλο (υπερπροσαρμογή).

Επομένως, η επιλογή των h και λ μπορεί να επηρεάσει σημαντικά το συμβιβασμό μεταξύ μεροληψίας (υποπροσαρμογή) και διακύμανσης (υπερπροσαρμογή), και αυτό θα είναι οπτικά εμφανές στις αλλαγές στο όριο απόφασης στο δισδιάστατο διάγραμμα.

### **ΑΠΟΤΕΛΕΣΜΑΤΑ**

Το νέο σημείο κατηγοριοποιείται ως κύκλος

Εικόνα που περιέχει κείμενο, διάγραμμα, γράφημα, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Το νέο σημείο κατηγοριοποιείται ως αστέρι

Εικόνα που περιέχει κείμενο, διάγραμμα, γράφημα, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

## **Πρόβλημα 3.3**

### **α)**

Ομαδοποίηση K-means

Ο αλγόριθμος ομαδοποίησης K-means είναι μια γνωστή τεχνική για την κατάτμηση των δεδομένων σε K συστάδες. Εδώ, τον εφαρμόσαμε για να ομαδοποιήσουμε τα δεδομένα σε δύο συστάδες.

Στη συνέχεια, υπολογίζεται το ποσοστό σφάλματος συγκρίνοντας τις προβλεπόμενες ομαδοποιήσεις με τις πραγματικές ομαδοποιήσεις. Γνωρίζουμε τις πραγματικές ομαδοποιήσεις επειδή τα πρώτα 100 σημεία ανήκουν σε μια ομάδα και τα δεύτερα 100 σημεία ανήκουν σε μια άλλη. Το σφάλμα για κάθε ομάδα υπολογίζεται ως το ποσοστό των σημείων της συγκεκριμένης ομάδας που ταξινομήθηκαν εσφαλμένα.

Για να το υπολογίσουμε αυτό, χρησιμοποιούμε τον κανόνα των προβλεπόμενων ετικετών για κάθε ομάδα. Ο κανόνας είναι η πιο κοινή ετικέτα , και θεωρείται ότι είναι η σωστή, δεδομένου ότι η πλειοψηφία των σημείων ταξινομείται σωστά. Εάν η αποδιδόμενη ετικέτα δεν ταιριάζει με τον κανόνα, θεωρείται σφάλμα.

Αφού υπολογίσουμε τα σφάλματα για κάθε ομάδα, υπολογίζουμε το συνολικό ποσοστό σφάλματος αθροίζοντας τα σφάλματα και διαιρώντας τα με τον συνολικό αριθμό των σημείων δεδομένων. Εάν το ποσοστό σφάλματος είναι κοντά στο 0,5, αυτό σημαίνει ότι η απόδοση ομαδοποίησης είναι παρόμοια με την τυχαία μαντεψιά, γεγονός που υποδηλώνει ότι ο αλγόριθμος K-means δεν λειτουργεί καλά για τα συγκεκριμένα δεδομένα.

### **ΑΠΟΤΕΛΕΣΜΑΤΑ**

|  |  |  |
| --- | --- | --- |
| **2D σφάλμα ομάδας 1** | **2D σφάλμα ομάδας 2** | **2D συνολικό σφάλμα** |
| 0,23 | 0,43 | 0,33 |

Εικόνα που περιέχει στιγμιότυπο οθόνης, κείμενο, διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα

### **β)**

K-means σε 3D:

K-means είναι γνωστό ότι παράγει γραμμικά όρια μεταξύ των συστάδων. Εάν οι πραγματικές συστάδες στα δεδομένα έχουν μη γραμμικά όρια, ο K-means στην αρχική του μορφή μπορεί να μην αποδώσει καλά. Μια προσέγγιση για την αντιμετώπιση αυτού του προβλήματος είναι η προβολή των δεδομένων σε έναν χώρο υψηλότερων διαστάσεων όπου οι συστάδες μπορεί να διαχωρίζονται γραμμικά.

Στον κώδικά μας, προσθέτουμε μια τρίτη διάσταση στα δεδομένα, η οποία είναι η τετραγωνική ευκλείδεια νόρμα (ή η τετραγωνική απόσταση από την αρχή). Αυτό βασίζεται στην παρατήρηση ότι μια ομάδα σημείων είναι συγκεντρωμένη γύρω από την αρχή των αξόνων και προσθέτοντας την τετραγωνική απόσταση ως πρόσθετο χαρακτηριστικό, τα σημεία που βρίσκονται κοντά στην αρχή θα είναι πιο κοντά στο οριζόντιο επίπεδο στον τρισδιάστατο χώρο.

Αφού εκτελέσουμε το K-means σε αυτά τα τρισδιάστατα δεδομένα, υπολογίζουμε τα ποσοστά σφάλματος όπως ακριβώς στο μέρος (α), αλλά τώρα με τις νέες τρισδιάστατες ετικέτες. Εάν το ποσοστό σφάλματος μειώνεται σε σύγκριση με την περίπτωση 2D, αυτό υποδηλώνει ότι η προσθήκη της τρίτης διάστασης έχει βελτιώσει την απόδοση του K-means.

Τέλος, σχεδιάζουμε τα αποτελέσματα και της 3D ομαδοποίησης K-means.

### **ΑΠΟΤΕΛΕΣΜΑΤΑ**

|  |  |  |
| --- | --- | --- |
| **3D σφάλμα ομάδας 1** | **3D σφάλμα ομάδας 2** | **3D συνολικό σφάλμα** |
| 0,01 | 0,31 | 0,16 |

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

## **Κώδικας Προβλήματος 3.1**

% Number of random variables

N = 1000;

% Random variables uniformly distributed in [0, 1]

x\_rand = rand(N,1);

% Bandwidths

h\_values = [0.001, 0.01, 0.1, 1];

% Plotting range

x = -0.5:0.01:1.5;

% Create figure

figure;

% Define colors

colors = ['r', 'g', 'b', 'm'];

% Loop over bandwidths

for i = 1:length(h\_values)

h = h\_values(i);

% Initialize density estimate

f = zeros(size(x));

% Loop over random variables

for j = 1:N

% Gaussian kernel

f = f + (1/(sqrt(2\*pi\*h)))\*exp(-((x-x\_rand(j)).^2)/(2\*h^2));

end

% Normalize

f = f / N;

% Plot with specific color

plot(x, f, colors(i), 'DisplayName', ['h = ' num2str(h)]);

hold on;

end

% Labeling

xlabel('x');

ylabel('Density');

legend('show');

hold off;

## **Κώδικας Προβλήματος 3.2**

% Load the data

load('data32.mat');

% Assign numerical labels

labels = [ones(size(stars,1), 1); -ones(size(circles,1), 1)];

% Concatenate the data

data = [stars; circles];

% Gaussian kernel function

GaussianKernel = @(x, y, h) exp(-1/h \* sum((x - y).^2));

% Set the specific values for h and lambda

h = 0.1;

lambda = 1;

% Initialize kernel matrix

K = zeros(size(data,1));

% Compute kernel matrix

for i = 1:size(data,1)

for j = 1:size(data,1)

K(i,j) = GaussianKernel(data(i,:), data(j,:), h);

end

end

% Alpha

alpha = (K + lambda \* eye(size(K))) \ labels;

% Grid for contour plot

[x1\_range, x2\_range] = meshgrid(min(data(:,1)):0.01:max(data(:,1)), min(data(:,2)):0.01:max(data(:,2)));

% Compute the function values on the grid

Z = zeros(size(x1\_range));

for i = 1:size(x1\_range,1)

for j = 1:size(x1\_range,2)

x\_new = [x1\_range(i,j), x2\_range(i,j)];

for k = 1:size(data,1)

Z(i,j) = Z(i,j) + alpha(k) \* GaussianKernel(x\_new, data(k,:), h);

end

end

end

% Plot the decision boundary

figure;

contour(x1\_range, x2\_range, Z, [0 0], 'k', 'LineWidth', 2); hold on;

% Plot the data points

scatter(stars(:,1), stars(:,2), 'b', 'filled'); hold on;

scatter(circles(:,1), circles(:,2), 'g', 'filled'); hold on;

% Generate a random new point

x1\_min = min(data(:,1));

x1\_max = max(data(:,1));

x2\_min = min(data(:,2));

x2\_max = max(data(:,2));

x\_new = [x1\_min + (x1\_max - x1\_min) \* rand(), x2\_min + (x2\_max - x2\_min) \* rand()];

% New point categorization

phi\_hat\_x\_new = zeros(size(data,1), 1);

for k = 1:size(data,1)

phi\_hat\_x\_new(k) = alpha(k) \* GaussianKernel(x\_new, data(k,:), h);

end

if sum(phi\_hat\_x\_new) > 0

disp('New point is categorized as a "star"');

scatter(x\_new(1), x\_new(2), 'r', 'filled');

else

disp('New point is categorized as a "circle"');

scatter(x\_new(1), x\_new(2), 'm', 'filled');

end

% Format the plot

title(['Decision boundary (h=', num2str(h), ', λ=', num2str(lambda), ')']);

xlabel('x1');

ylabel('x2');

legend('Decision boundary', 'Stars', 'Circles', 'New Point');

## **Κώδικας Προβλήματος 3.3**

% Load data

loadedData = load('data33.mat');

% The loaded data is a struct; we need to extract the matrix from it.

fieldnames = fields(loadedData);

data = loadedData.(fieldnames{1});

% Transpose data to have 200 rows and 2 columns

data = data';

% Check if data loaded correctly

if size(data, 1) < 200 || size(data, 2) < 2

error('Not enough data points. Please use a dataset with at least 200 data points.')

end

% K-means clustering

k = 2;

max\_iterations = 100;

[Idx, ~] = myKmeans(data, k, max\_iterations);

% Calculate error rates

error1 = sum(Idx(1:100) ~= mode(Idx(1:100)));

error2 = sum(Idx(101:end) ~= mode(Idx(101:end)));

kmeans\_error\_rate = (error1 + error2) / size(data, 1);

% Display 2D error rates

disp(['2D Error rate for group 1: ', num2str(error1/100)]);

disp(['2D Error rate for group 2: ', num2str(error2/100)]);

disp(['2D Total error rate: ', num2str(kmeans\_error\_rate)]);

% Create 3D data

data3D = [data, vecnorm(data, 2, 2).^2];

% K-means clustering in 3D

[Idx3D, ~] = myKmeans(data3D, k, max\_iterations);

% Calculate error rates in 3D

error1\_3D = sum(Idx3D(1:100) ~= mode(Idx3D(1:100)));

error2\_3D = sum(Idx3D(101:end) ~= mode(Idx3D(101:end)));

kmeans\_error\_rate\_3D = (error1\_3D + error2\_3D) / size(data, 1);

% Display 3D error rates

disp(['3D Error rate for group 1: ', num2str(error1\_3D/100)]);

disp(['3D Error rate for group 2: ', num2str(error2\_3D/100)]);

disp(['3D Total error rate: ', num2str(kmeans\_error\_rate\_3D)]);

% Plot 2D clusters

figure;

gscatter(data(:, 1), data(:, 2), Idx);

title('K-means Clustering in 2D');

xlabel('X');

ylabel('Y');

% Plot 3D clusters

figure;

scatter3(data3D(:, 1), data3D(:, 2), data3D(:, 3), 15, Idx3D, 'filled');

title('K-means Clustering in 3D');

xlabel('X');

ylabel('Y');

zlabel('Z');

% K-means function

function [idx, centroids] = myKmeans(data, k, max\_iterations)

[m, n] = size(data);

centroids = data(randperm(m, k), :);

idx = zeros(m, 1);

for iteration = 1:max\_iterations

old\_centroids = centroids;

% Assign each data point to the closest centroid

for i = 1:m

[~, idx(i)] = min(vecnorm(data(i,:) - centroids, 2, 2));

end

% Recalculate centroids

for i = 1:k

centroids(i, :) = mean(data(idx == i, :), 1);

end

if old\_centroids == centroids

break;

end

end

end