Ενδιάμεση Εργασία στα Νευρωνικά Δίκτυα

ΑΝΔΡΕΑ ΣΕΓΚΑΝΙ ΑΕΜ 10770

Σε αυτή την εργασία, στόχος ήταν η σύγκριση των κατηγοριοποιητών **K-Nearest Neighbors (KNN)** με 1 και 3 πλησιέστερους γείτονες, με τον κατηγοριοποιητή **K-Centroids** στην **βάση δεδομένων CIFAR-10**. Το πρόβλημα επικεντρώνεται στην ταξινόμηση εικόνας με χρήση των δύο αλγορίθμων, και η σύγκριση γίνεται βάσει της ακρίβειας που επιτυγχάνουν οι δύο κατηγοριοποιητές.

Ο αλγόριθμος που χρησιμοποιήθηκε ήταν ο εξής:

$$\text{distance(x,Xi)}=\sqrt{\sum_{j=1}^{d}(xi - Xij)}$$

Αρχικά υπολογίστηκε η ευκλείδεια απόσταση μεταξύ των σημείων train και όλων του test.

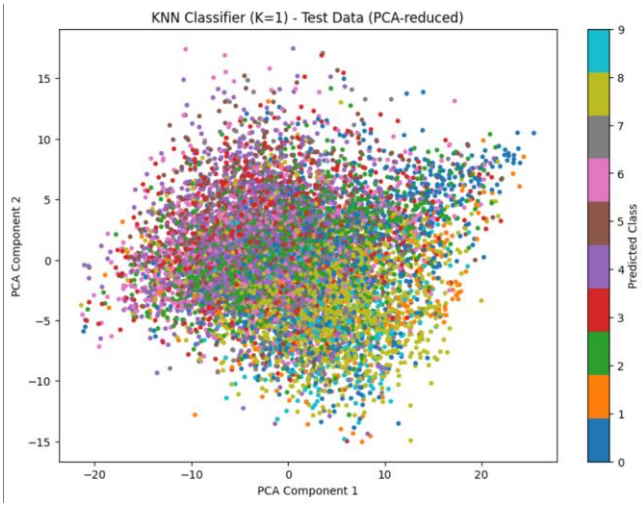
- Σύγκριση με τις αποστάσεις των γειτόνων: Αν η απόσταση που υπολογίστηκε είναι μικρότερη από την απόσταση κάποιου γείτονα στον πίνακα των μικρότερων αποστάσεων (DistanceToNN), τότε αφαιρούμε τον γείτονα αυτόν και προσθέτουμε το νέο δείγμα με την αντίστοιχη απόσταση, ώστε να διατηρηθεί ο πίνακας των k κοντινότερων γειτόνων.

- **Πρόβλεψη:** Όταν έχουμε τα k πιο κοντινά δείγματα, παίρνουμε την **πιο συχνή κατηγορία** από αυτά και αυτή είναι η πρόβλεψη του αλγορίθμου για το δείγμα δοκιμής.

- Τα δεδομένα εκπαίδευσης και δοκιμής μετατρέπονται σε **1D πίνακες** για να μπορούν να συγκριθούν.

- Στο τέλος, προβλέπεται η ετικέτα για το πρώτο δείγμα δοκιμής και εμφανίζεται το αποτέλεσμα.

Ο αλγόριθμος **Nearest Centroid** είναι η άλλη απλή μέθοδος κατηγοριοποίησης. Αντί να κοιτάει τα κοντινότερα δείγματα όπως ο KNN, υπολογίζει το κέντρο κάθε κατηγορίας (τον μέσο όρο των χαρακτηριστικών των δειγμάτων αυτής της κατηγορίας). Όταν χρειάζεται να κατηγοριοποιήσει ένα νέο δείγμα, το αντιστοιχεί στην κατηγορία του κοντινότερου κέντρου. Αυτή η μέθοδος είναι γρήγορη και δεν απαιτεί υπολογισμούς για κάθε δείγμα, όπως ο knn.



Plot των σημείων σε δισδιάστατο χώρο.

A. K=1

Ο αλγόριθμος επιλέγει τον πλησιέστερο γείτονα από το σύνολο εκπαίδευσης και προβλέπει την ετικέτα του.
Τα αποτελέσματα αυτού είναι τα εξής:

```
# Κατηγοριοποιητής 1-Nearest Neighbor (K=1)
knn_1 = KNeighborsClassifier(n_neighbors=1)
knn_1.fit(train_data, train_labels)
predictions_knn_1 = knn_1.predict(test_data)
accuracy_knn_1 = accuracy_score(test_labels, predictions_knn_1)
print("Ακρίβεια κατηγοριοποιητή KNN με K=1:", accuracy_knn_1)

Ακρίβεια κατηγοριοποιητή KNN με K=1: 0.3539
```

Όπως βλέπουμε χρησιμοποιήθηκε το έτοιμο και βελτιστοποιημένο μοντέλο για το prediction έχοντας ως ακρίβεια 0.3539. ΒΑΛΕ ΤΗΝ ΑΠΟΣΤΑΣΗ

B. K=3

Ο αλγόριθμος κοιτάζει τους **3 κοντινότερους γείτονες** και επιστρέφει την ετικέτα που εμφανίζεται περισσότερες φορές από αυτούς.
Τα αποτελέσματα αυτού είναι τα εξής:

```
[29] knn_3 = KNeighborsClassifier(n_neighbors=3)
     knn_3.fit(train_data, train_labels)
     predictions_knn_3 = knn_3.predict(test_data)
     accuracy_knn_3 = accuracy_score(test_labels, predictions_knn_3)
     print("Ακρίβεια κατηγοριοποιητή KNN με K=3:", accuracy_knn_3)

     Ακρίβεια κατηγοριοποιητή KNN με K=3: 0.3303
```

Όπως βλέπουμε και στη περίπτωση αυτή χρησιμοποιήθηκε το έτοιμο και βελτιστοποιημένο μοντέλο για το prediction έχοντας ως ακρίβεια 0.3303.Επίσης σε αυτή τη περίπτωση δεν χρησιμοποιήσαμε την ευκλείδεια απόσταση αλλά
Με το k=3, το μοντέλο έχει περισσότερους γείτονες να εξετάσει, οπότε μπορεί να γίνει λιγότερο ακριβές, ειδικά αν κάποιοι από αυτούς τους γείτονες είναι λιγότερο σχετικοί.

Γ. Αυτοσχέδια εκδοχή για k=3

Σε αυτή τη περίπτωση δημιουργήθηκε μια αυτοσχέδια εκδοχή του 3-nn η οποία όμως είχε πολύ μεγάλο χρόνο runtime οπότε δεν προτείνεται για εκτέλεση .

```
def euclidean_distance(x1, x2):
    return np.sqrt(np.sum((x1 - x2) ** 2)) # Αρχικά υπολογίζεται η ευκλείδεια απόσταση δυο σημείων που θα αξιοποιηθεί για το train μας.

def custom_knn(test_example, train_data, train_labels, k=3): #Φτιάχνουμε το δικό μας knn algorithm και αρχικοποιούμε τους πινακες κοντινης αποστασης και ετικέτας και επιλέγουμε
    nearest_distances = []
    nearest_labels = []

    for i, train_example in enumerate(train_data): #Παρατηρούμε την απόσταση κάθε στοιχείου του train για όλα τα στοιχεία του test και την αποθηκεύουμε στο dist
        dist = euclidean_distance(test_example, train_example)


        if len(nearest_distances) < k: # Αν ο πινακας με τις ελαχιστες αποστασεις ειναι μικροτερος απο τους γειτονες που θελουμε ξεκινμε τη διαδικασια
            nearest_distances.append(dist) # αποθηκευουμε τις πιο κοντινες αποστασεις καθως και τα labels τους
            nearest_labels.append(train_labels[i])
        else:
            max_dist_index = np.argmax(nearest_distances)
            if dist < nearest_distances[max_dist_index]:
                nearest_distances[max_dist_index] = dist
                nearest_labels[max_dist_index] = train_labels[i]


    counts = np.bincount(nearest_labels)  # πίνακας που επιστρέφει τη κατανομή των  συχνοτήτων των γειτονων
    most_common_label = np.argmax(counts)  # κανουμε get εκεινο το γειτονα με τη μεγαλυτερη συχνοτητα
    return most_common_label


train_data_flat = train_data.reshape(len(train_data), -1)
test_data_flat = test_data.reshape(len(test_data), -1)

# Example usage for a single test example
sample_test_example = test_data_flat[0]  # Example test sample
predicted_label = custom_knn(sample_test_example, train_data_flat, train_labels, k=3)
print("Predicted label for test example:", predicted_label)
```

Στον κώδικα υπάρχουν και σχόλια για κάθε βήμα που δείχνουν τη σκέψη για τον αλγόριθμο.

```
def calculate_accuracy(test_data, test_labels, train_data, train_labels, k=3):
    correct_predictions = 0
    for i in range(len(test_data)):
        test_example = test_data[i]
        true_label = test_labels[i]

        predicted_label = custom_knn(test_example, train_data, train_labels, k)

        if predicted_label == true_label:
            correct_predictions += 1

    accuracy = correct_predictions / len(test_data) * 100
    return accuracy


accuracy_knn_3 = calculate_accuracy(test_data_flat, test_labels, train_data_flat,
print(f"Ακρίβεια για custom K-NN classifier with K=3: {accuracy_knn_3:.2f}%")
```

Τα αποτελέσματα του δικού μου αλγορίθμου είναι

```
print("Ακρίβεια για custom K-NN classifier with K=3:",

Ακρίβεια για custom K-NN classifier with K=3: 0.2994
```

Δ. Nearest Centroid

```
nearest_centroid = NearestCentroid()
nearest_centroid.fit(train_data, train_labels)
predictions_centroid = nearest_centroid.predict(test_data)
accuracy_centroid = accuracy_score(test_labels, predictions_centroid)
print("Ακρίβεια κατηγοριοποιητή Nearest Centroid:", accuracy_centroid)

Ακρίβεια κατηγοριοποιητή Nearest Centroid: 0.2774
```

Όπως βλέπουμε από τα αποτελέσματα τρέχει πολύ πιο γρήγορα από τις άλλες μεθόδους έχοντας όμως τη μικρότερη απόδοση (0.2274). Αυτό συμβαίνει διότι οι κατηγορίες είναι ανομοιογενείς και η μέθοδος αυτή δεν είναι τόσο αξιόπιστη ,καθώς τα δείγματα μιας κατηγορίας είναι διασκορπισμένα ή περιέχουν μεγάλα αποστάσεις μεταξύ τους, το κέντρο μπορεί να μην αντιπροσωπεύει καλά την κατηγορία, όπως φαίνεται και πάνω στο plot.