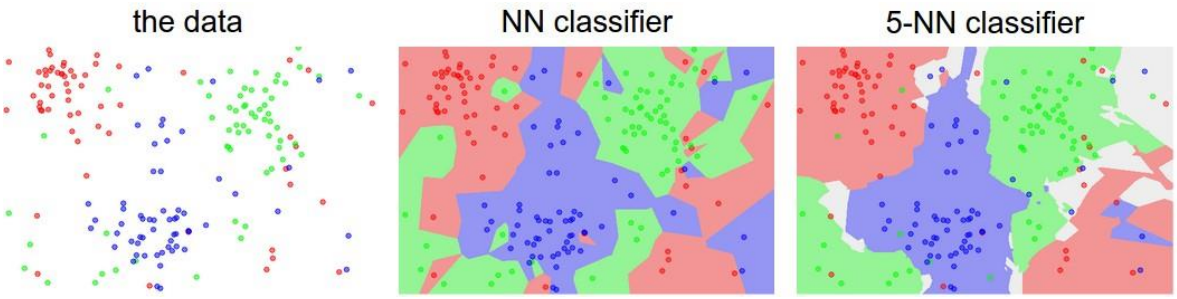
Metoda celor mai apropiați vecini



Exemplu care arată diferențele dintre metoda celui mai apropiat vecin și metoda celor mai apropiați cinci vecini. Zona colorată reprezintă regiunea de decizie a clasificatorului folosind distanța L2. Se observă că în cazul metodei celui mai apropiat vecin se formează mici 'insule' ce pot duce la predicții incorecte. Zonele gri din imaginea 5-NN reprezintă zone de predicție ambigue din cauza egalității voturilor celor mai apropiați vecini.

În acest laborator vom clasifica cifrele scrise de mana din subsetul **MNIST** folosind metoda celor mai apropiați vecini.

Descărcați arhiva cu datele de antrenare și testare de aici.

❓ Care este acuratețea metodei *celui* mai apropiat vecin pe mulțimea de *antrenare* când se folosește distanța L2? Dar pentru distanța L1?

❓ Care este acuratețea metodei celor mai apropiați vecini pe mulțimea de *antrenare* când se folosește numărul de vecini $K \geq 2$ și distanța L2? Dar pentru distanța L1?

Exerciții

1. Creați clasa Knn_classifier, având constructorul următor:

```
def __init__(self, train_images, train_labels):
    self.train_images = train_images
    self.train_labels = train_labels
```

2. Definiți metoda *classify_image(self, test_image, num_neighbors = 3, metric = 'l2')* care clasifică imaginea *test_image* cu metoda celor mai apropiați vecini, numărul vecinilor este stabilit de parametru *num_neighbors*, iar distanța poate fi L1 sau L2, în funcție de parametrul *metric*.
   Obs:
   - $L1(X,Y) = \sum_{i=1}^{n} |X_i - Y_i|$

- $L2(X,Y) = \sqrt{\sum_{i=1}^{n} (X_i - Y_i)^2}$
- În variabilele *train_images* și *test_image* valorile unui exemplu sunt stocate pe linie. (train_images.shape = (num_samples, num_features), test_image.shape = (1, num_features))

3. Calculați acuratețea metodei celor mai apropiați vecini pe mulțimea de testare având ca distanța 'l2' și numărul de vecini 3. Salvați predicțiile în fișierul *predictii_3nn_l2_mnist.txt*.
   Obs:
   - Acuratețea pe mulțimea de testare este de 89.8%.

4. Definiți metoda *confusion_matrix(y_true, y_pred)* care calculează matricea de confuzie. Calculați matricea de confuzie folosind predicțiile din *predictii_3nn_l2_mnist.txt.*
   Obs:
   - Matrice de confuzie $C = c_{ij}$, numărul exemplelor din clasa *i* care au fost clasificata ca fiind în clasa *j*.

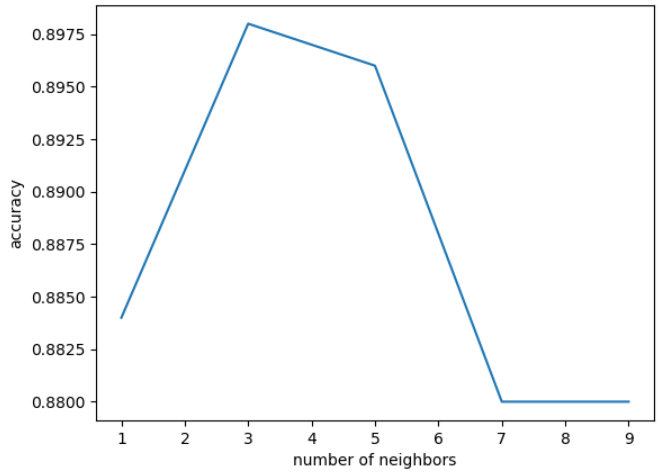
Clasa actuală↓ Clasa prezisă →	1	2	3
1	Nr. exemplelor din clasa 1 care au fost clasificate ca fiind in clasa 1	Nr. exemplelor din clasa 1 care au fost clasificate ca fiind in clasa 2	Nr. exemplelor din clasa 1 care au fost clasificate ca fiind in clasa 3
2	Nr. exemplelor din clasa 2 care au fost clasificate ca fiind in clasa 1	Nr. exemplelor din clasa 2 care au fost clasificate ca fiind in clasa 2	Nr. exemplelor din clasa 2 care au fost clasificate ca fiind in clasa 3
3	Nr. exemplelor din clasa 3 care au fost clasificate ca fiind in clasa 1	Nr. exemplelor din clasa 3 care au fost clasificate ca fiind in clasa 2	Nr. exemplelor din clasa 3 care au fost clasificate ca fiind in clasa 3

   - Matricea de confuzie pentru clasificatorul anterior este:

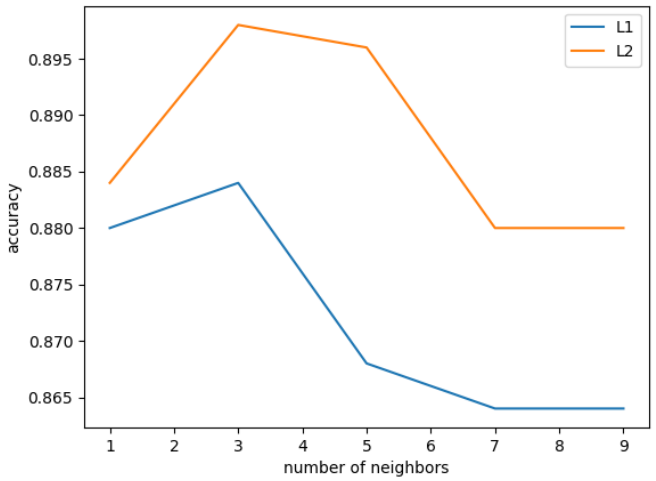
```
[[51.  0.  0.  0.  0.  1.  1.  0.  0.  0.]
 [ 0. 52.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 1.  6. 47.  1.  0.  0.  1.  2.  0.  0.]
 [ 0.  0.  0. 51.  0.  1.  0.  0.  0.  1.]
 [ 0.  0.  0.  0. 44.  0.  0.  0.  0.  2.]
 [ 2.  1.  1.  6.  0. 40.  1.  0.  0.  1.]
 [ 0.  0.  0.  0.  0.  1. 47.  0.  0.  0.]
 [ 1.  2.  0.  0.  1.  0.  0. 46.  0.  0.]
 [ 1.  0.  2.  2.  1.  1.  1.  1. 36.  1.]
 [ 0.  0.  1.  1.  3.  1.  0.  1.  0. 35.]]
```

5. Calculați acuratețea metodei celor mai apropiați vecini pe mulțimea de testare având ca distanța L2 și numărul de vecini ∈ [1, 3, 5, 7, 9].

a. Plotați un grafic cu acuratețea obținuta pentru fiecare vecin și salvați scorurile în fișierul *acuratete_l2.txt*.



b. Repetați punctul anterior pentru distanța L1. Plotați graficul de la punctul anterior în aceeași figură cu graficul curent (utilizați fișierul *acuratete_l2.txt*).



Funcții numpy:

```
np.sort(x)  # sorteaza array-ul
np.argsort(x)  # returneaza indecsi care sorteaza array-ul
np.bincount(x) # calculeaza numarul de aparatii al fiecarei valori din array
print(np.bincount(numpy.array([0, 1, 1, 3, 2, 1, 7]))) # array([1, 3, 1, 1, 0, 0, 0, 1])
np.where(x == 3) # returneaza indecsi care satisfac conditia
np.intersect1d(x, y) # returneaza intersectia celor 2 array
np.savetxt('fisier.txt', y) # salveaza array-ul y in fisierul fisier.txt
```