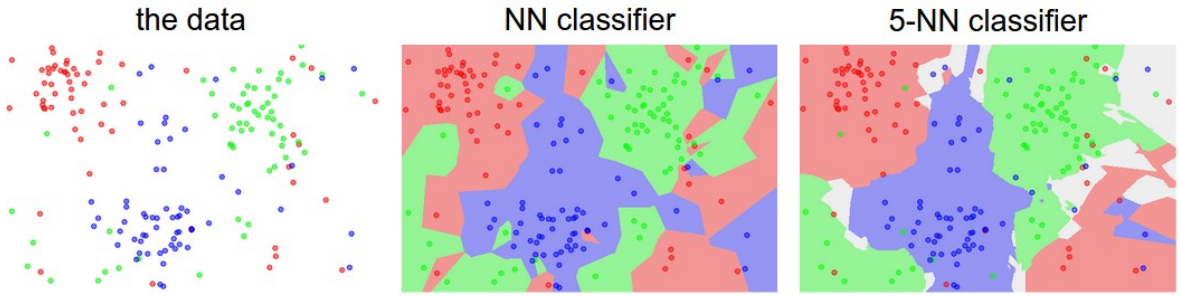
Metoda celor mai apropiați k-vecini



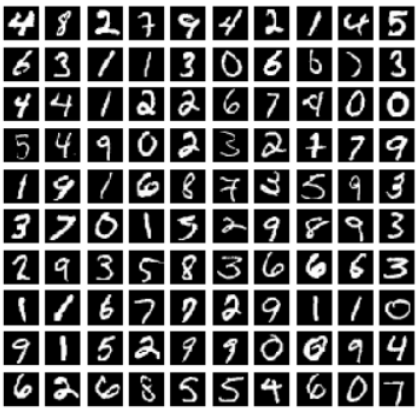
Exemplu care arată diferențele dintre metoda celui mai apropiat vecin și metoda celor mai apropiați cinci vecini. Zona colorată reprezintă regiunea de decizie a clasificatorului folosind distanța euclidiană l_2. Se observă că în cazul metodei celui mai apropiat vecin se formează mici 'insule' ce pot 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 mână din subsetul **MNIST** folosind metoda celor mai apropiați k-vecini.

MNIST[1] este o bază de date cu cifre scrise de mână (0-9), conținând 60.000 de imagini pentru antrenare și 10.000 pentru testare. Imaginile sunt în tonuri de gri (grayscale) având dimensiunea de 28×28 pixeli. În cadrul laboratorului vom lucra pe un subset, împărțit astfel:

- în fișierul 'train_images.txt' sunt 1.000 de exemple (imagini) din mulțimea de antrenare, fiecare exemplu de antrenare fiind stocat pe câte o linie a matricei de dimensiune 1000 x 784 (28 x 28 = 784).
- în fișierul 'test_images.txt' sunt 500 de exemple (imagini) din mulțimea de testare.
- fișierele 'train_labels.txt' și 'test_labels.txt' conțin etichetele exemplelor de antrenare respectiv testare.

Figura de mai jos afișează primele 100 exemple din mulțimea de testare (stânga) împreună cu etichetele lor (dreapta).



```
[[4 8 2 7 9 4 2 1 4 5]
 [6 3 1 1 3 0 6 6 7 3]
 [4 4 1 2 2 6 7 4 0 0]
 [5 4 9 0 2 3 2 7 7 9]
 [1 9 1 6 8 7 3 5 9 3]
 [3 7 0 1 3 2 9 8 9 3]
 [2 9 3 5 8 3 6 6 6 3]
 [1 1 6 7 7 2 9 1 1 0]
 [9 1 5 2 9 9 0 0 9 4]
 [6 2 6 8 5 5 4 6 0 7]]
```

Descărcați fisierul data.zip cu datele de antrenare și testare de pe Moodle.

[1] http://yann.lecun.com/exdb/mnist/

 Care este acuratețea metodei *celui* mai apropiat vecin pe mulțimea de *antrenare* când se folosește distanța l_2? Dar pentru distanța l_1?

Exerciții

1. Considerați primul exemplu din mulțimea de testare (este o imagine cu cifra 2). Determinați și afișați (plotând într-o figură) cei mai apropiați k-vecini ai acestui exemplu de testare din mulțimea de antrenare folosind distanța euclidiană (l_2). Folosiți valorile k = 1, 3, 5, 7. Care va fi eticheta asignată exemplului de testare pentru fiecare din cele 4 cazuri?

2. Creați clasa Knn_classifier, având constructorul următor:

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

3. Definiți pentru clasa de mai sus 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 l_1 (distanța Manhattan) sau l_2 (distanța euclidiană), în funcție de parametrul *metric*.

   Observație:
   - pentru vectorii $\boldsymbol{x} = (x_1, x_2, \dots x_n)$ și $\boldsymbol{y} = (y_1, y_2, \dots y_n)$ distanțele l_1 și l_2 se definesc astfel:
     $l_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} |x_i - y_i|$, $l_2(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}$
   - 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))

4. Calculați acuratețea metodei celor mai apropiați vecini pe mulțimea de testare având ca distanță l_2 și numărul de vecini 3. Salvați predicțiile în fișierul *predictii_3nn_l2_mnist.txt*.
   Observație:
   - Acuratețea pe mulțimea de testare este de 89.8%.

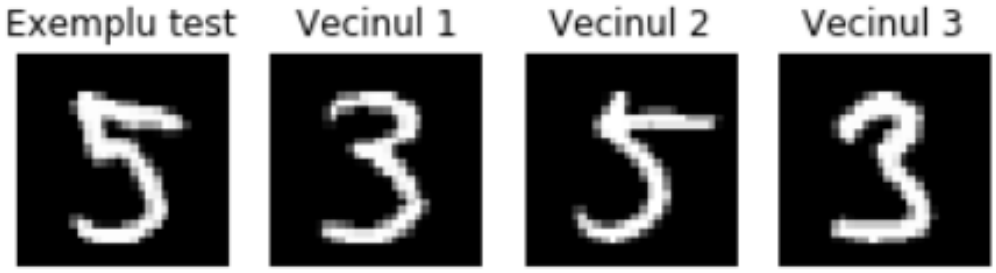
5. Definiți metoda *confusion_matrix(y_true, y_pred)* care calculează matricea de confuzie. Calculați matricea de confuzie folosind predicțiile din *predictii_3nn_l2_mnist.txt.*
   Observație:
   - Pentru matricea de confuzie *C*, fiecare element c_{ij} reprezintă numărul exemplelor din clasa *i* care au fost clasificate ca fiind în clasa *j*.

Clasa prezisă → Clasa actuală↓	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.]]
```

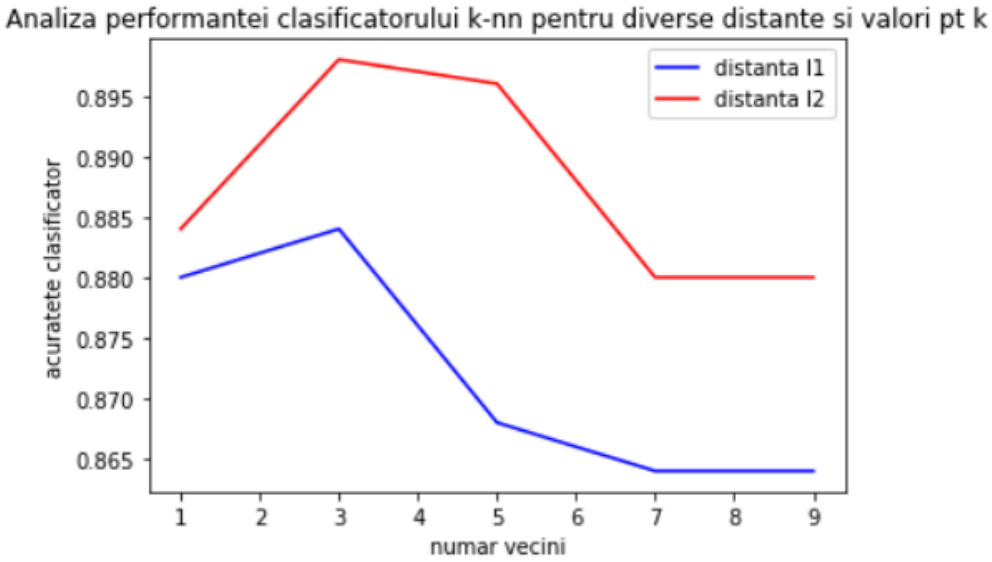
6. Perechea de cifre (5, 3) cea mai des confundată, având șase misclasificări. Afișați exemplele de cifra 5 misclasificate precum și cei trei vecini. Mai jos este afișat primul din cele șase astfel de cazuri.



7. Calculați acuratețea metodei celor mai apropiați vecini pe mulțimea de testare având ca distanța l_2 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 l_1. Plotați graficul de la punctul anterior în aceeași figură cu graficul curent (utilizați fișierul *acuratete_l1.txt*).

Ar trebui să obțineți o figură similară cu cea de mai jos:



Funcții utile din 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
```