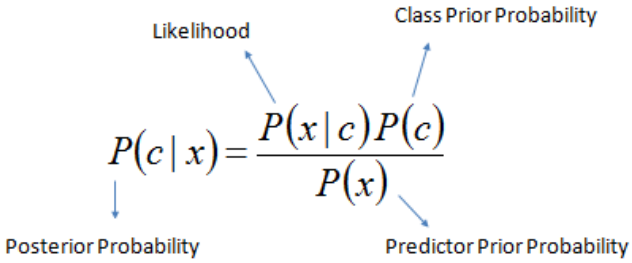
Naive Bayes

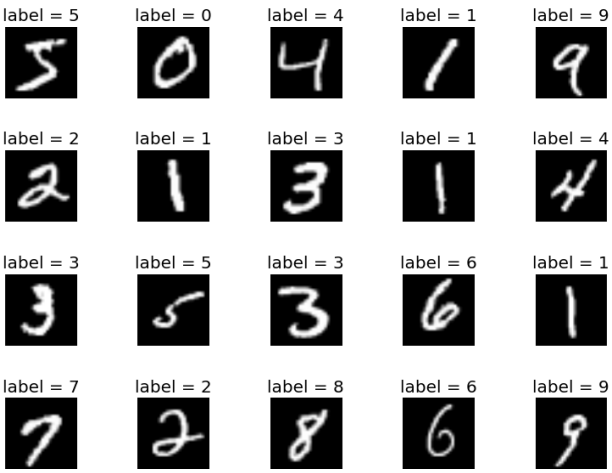


Regula Bayes

În acest laborator vom clasifica cifrele scrise de mână din subsetul **MNIST** folosind Naive Bayes.

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 alb-negru având dimensiunea de 28x28 pixeli. În cadrul laboratorului vom lucra pe un subset, împărțit astfel:

- ➔ În 'train_images.txt' sunt 1.000 de imagini din mulțimea de antrenare, fiecare fiind stocată pe câte o linie a matricei de dimensiune 1000 x 784 (28 x 28 = 784).
- ➔ În 'test_images.txt' sunt 500 de imagini din setul de testare.
- ➔ Fișierele 'train_labels.txt' și 'test_labels.txt' conțin etichetele imaginilor.



Exemple de imagini din setul de date MNIST.

Descărcați arhiva care conține datele de antrenare și testare de aici.

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

Pentru vizualizarea unei imagini din mulțimea de antrenare trebuie să redimensionăm vectorul de 1 x 784 la 28 x 28.

```
from skimage import # pentru afisarea imaginii

train_images = np.loadtxt('train_images.txt') # incarcam imaginile
train_labels = np.loadtxt('train_labels.txt', 'int')  # incarcam etichetele avand
                                                      # tipul de date int
image = train_images[0, :] # prima imagine
image = np.reshape(image, (28, 28))
io.imshow(image.astype(np.uint8))
io.show()
```

Deoarece datele noastre (valorile pixelilor) sunt valori continue, va trebui sa le transformăm în valori discrete cu ajutorul unei histograme. Vom stabili numărul de intervale la care vom împarți lungimea intervalului valorilor continue, apoi vom asigna fiecărei valori continue indicele intervalul corespunzător.

```
bins = np.linspace(start=0, stop=255, num=num_bins)  # returneaza intervalele
x_to_bins = np.digitize(x, bins) # returneaza pentru fiecare element intervalul
                                 # corespunzator
                                 # Atentie! Indexarea primul interval se face de la 1
```

1. Antrenarea clasificatorului (fit)

O imagine $X = \{x_1, x_2, \ldots, x_{784}\}$ din mulțimea de antrenare are dimensiunea de 1x784. Conform presupunerii clasificatorului Naive Bayes, vom considera fiecare pixel un atribut ***independent*** în calcularea probabilității apartenenței lui *X* la clasa *c*.

$$P(c|X) = P(c|x_1) * P(c|x_2) *\ldots* P(c|x_{784}) \quad | \; aplicam\ logaritm$$
$$log(P(c|X)) = log(P(c|x_1)) + log(P(c|x_2)) + \ldots + log(P(c|x_{784}))$$

Pentru aplicarea regulii Naive Bayes avem nevoie de:

1. $P(c)$ $= probabilitatea\ ca\ un\ exemplu\ să\ se\ afle\ în\ clasa\ c$
2. $P(x)$ $= probabilitatea\ atributului\ x$
3. $P(x|c)$ $= probabilitatea\ de\ a\ avea\ atributul\ x\ în\ clasa\ c$

Pentru implementarea regulii Naive Bayes avem nevoie de:

1. Un vector de dimensiune (num_classes, 1) în care să stocăm *P(c)* (probabilitatea fiecărei clase).
2. Un vector de dimensiune (num_features, num_bins), în care pentru fiecare atribut și pentru fiecare interval stocăm *P(x)* (probabilitatea atributului x).
3. Un vector de dimensiune (num_features, num_bins, num_classes), în care stocăm *P(x | c) (*probabilitatea de a avea atributul x în clasa c).

2. Prezicerea etichetelor pe baza clasificatorul (predict)

$$P(c|X) = \prod_{i=1}^{n} \frac{P(x_i|c) * P(c)}{P(x_i)}, unde\ X = \{x_1, x_2, \dots, x_n\}\ cu\ x_1, \dots, x_n\ atribute\ independente.$$

Probabilitatea ca exemplul $X = \{x_1, x_2, \dots, x_{784}\}$ să fie in clasa c se obține prin înmulțirea (sau adunarea logaritmilor) probabilităților individuale ale atributelor acestuia condiționate de clasa c. Vom calcula P(c|X) pentru fiecare clasa c (c∈ [1, num_classes]), iar eticheta finală este dată de clasa cu probabilitatea cea mai mare.

Pentru implementarea predicției folosind clasificatorul Naive Bayes avem nevoie de:

1. Cei 3 vectori (*P(x), P(c), P(x|c)*) calculați anterior.
2. Un vector *probs* de dimensiune (num_samples, num_classes) în care să stocăm *P(c|X)* (probabilitatea ca *X* să se afle în clasa *c*) pentru fiecare exemplu de test.
3. Eticheta prezisă este indicele probabilității maxime de pe fiecare linie din vectorul *probs*.

Exerciții

1. Creați clasa **NaiveBayes**, având constructorul *__init__(self, num_bins, max_value)* în care se setează și se calculează extremitățile intervalelor. Definiți metoda *values_to_bins* care primește o matrice de dimensiune (n_samples, n_features), iar pentru fiecare exemplu și fiecare atribut calculează indexul intervalului corespunzător.

2. Definiți metoda *fit* care primește datele de antrenare (*train_images* și *train_labels)* și calculează și stochează probabilitățile necesare aplicării regulii Naive Bayes (după formulele de mai sus).

   OBS. Pentru stabilitate numerică adunați 1e-10 la *P(x)* și *P(x|c)*.

3. Definiți metoda *predict* care primește argumentul *test_images* și returnează etichetele prezise folosind regula Naive Bayes (logaritmând probabilitățile).

4. Definiți metoda *score* care primește argumentele *test_images* și *test_labels* și returnează acuratețea medie.

5. Testați clasificatorul Naive Bayes pe subsetul MNIST folosind$num_bins \in \{3, 5, 10\}$.

OBS. Acuratețea pe care trebuie să o obțineți pentru num_bins = 5 este de 70.4%.

6. Probabilitatea de a alege un număr între 0 și 9 este de 0.1, dar din cauză că avem o mulțime de antrenare mică, probabilitățile claselor (numerelor de la 0 la 9) nu sunt egale. Suprascrieți probabilitățile claselor, astfel încât să aveți probabilități egale și testați clasificatorul pentru num_bins = 3. Ce observați?

Funcții numpy:

```
x = np.array([1, 2, 3, 4, 3, 4])
np.argmin(x)  # returneaza pozitia elementului minim
np.argmax(x)  # returneaza pozitia elementului maxim
np.where(x == 3) # returneaza indecsi care satisfac conditia
```