Documentaţie pentru proiectul de la ML - Mihalea Andreas, grupa 311

Cap 1: Noţiuni introductive

1. Noţiunea de normă pe spaţiul $\mathbb{R}^n$.

Fie $\|\|_k : \mathbb{R}^n \to \mathbb{R},\ \|x\|_k = (\sum_{i=1}^n x_i^k)^{\frac{1}{k}}$, in particular $\|\|_\infty : \mathbb{R}^n \to \mathbb{R},\ \|x\|_\infty = \max_{1 \leq i \leq n} |x_i|$.

Noi o să folosim norma infinită in a determina cea mai mare acurateţe (nu in procentaj) indiferent de tipul de model şi norma $\|\|_2$ pentru a putea determina un mod de detectare a pierderii

2. Noţiunea de transformare liniară

Fie $f : \mathbb{R}^n \to \mathbb{R}^n$, unde $f(x+y) = f(x) + f(y), \forall \alpha \in \mathbb{R}^n,\ \alpha f(x) = f(\alpha x), \forall x \in \mathbb{R}^n$, in particular $f(x) = xA' + b, \forall x \in R^n, A'$ - matrice transpusă. Vom aplica această notiune pentru regresia liniară aplicată pentru Convolutional Neural Network.

3. Noţiunea de gradient pe spaţiul $\mathbb{R}^n$

Vom nota $\nabla f(x) = \left(\frac{\partial f(x)}{x_1}, \frac{\partial f(x)}{x_2}, ..., \frac{\partial f(x)}{x_n}\right), \forall x \in \mathbb{R}^n$.

Gradientul este folosit pentru optimizare.

4.Funcţia "Loss"

Vom defini funcţia "Loss" notată cu L astfel:

$$L : \mathbb{R}^n \to \mathbb{R}, L(x) = -x_{class} + \sum_{i=1}^{n} e^{x_i}, \forall x \in \mathbb{R}^n$$

Este mereu folosit pentru a determina pierderea imaginii, cu cat e mai mică pierderea cu atat este mai ok modelul.

5. Noţiunea de matrice de confuzie.

Fie $M \in \mathbb{R}^{3\times 3}, M$ - matrice de confunzie a.e
$M = \{m_{ii} =$ relatia de echivalenta dintre prezicere si exactiate$, m_{ii} \neq 0, m_{ij} =$ altfel $, \forall i \neq j \in \{1, 2, 3\}\}$, ce arată in prin plan performanţa modelului selectat.

6. Eroarea relativă

Fie $x \in \mathbb{R}^n$, vom defini ca:

$$e_r : \mathbb{R}^n \to \mathbb{R}, e_r(x) = \left\| \frac{x - x_{corect}}{\max\{x, x_{corect}\} * 100} \right\|_\infty$$

Acurateţia o voi determina folosind $a_r = 1 - e_r$

Cap 2: Preprocesare date din fişiere.

Prima dată vom citi datele din fiserele text, train.txt si din validation.txt, şi le vom băga intr-un dicţionar. Apoi voi citi fişierele din folderele ”train” şi ”validation” şi le voi incărca.
Fiecare imagine este normalizată cu o medie de 42.7 şi cu-n standard de 47.5.

```
def load_list_of_data(self):
    #citesc din fisierele text: nume si denumire, mi-l baga intr-un dictionar ai dictionar[filename] = 0, 1 sau 2
    self.dictionary = {}
    f = open(self.text_file, "r")
    content = f.read()
    for line in content.split("\n"):
        line_data = line.split(",")
        if len(line_data) == 2: self.dictionary[line_data[0]] = line_data[1]
    return self

def load_images(self, folder):
    #citesc imaginiile si atribui intr-un tensor, o combinatie de forma [1, 50, 50], type ce poate avea valoarea de la 0, 1 sau 2
    images = [normalize(to_tensor(cv2.imread(os.path.join(folder, filename), 0)), [mean], [std]) for filename in os.listdir(folder)]
    type = [int(self.dictionary[filename]) for filename in os.listdir(folder)]
    return torch.unsqueeze(torch.cat(images, dim=0), dim=1), torch.Tensor(type)
```

Cap 3: Modele incercate:

1. Convolutional Neural Networks

Eu am facut in mod general, n layere aranjate intr-un mod liniar, cu un vector de conventii şi de caracteristici.

```
def __init__(self, n = 3, channels_conv = [1, 1024, 512, 512], features = [8192, 2048, 1024, 1024], kernel_size = (5, 5), padding = (1, 1)):
    #clasa neuralnetwork
    #self.n reprezinta cate layere doresc sa am
    #channels_conv reprezinta conventiile pentru fiecare layer in parte
    #features reprezinta numarul de noduri din CNN
    #kernel_size reprezinta dimensiunea convolutiei, default (5, 5)
    #padding iti extinde sa zicem imaginea pentru a nu pierde datele din colturi (la inceput nu e nevoie)
    #am considerat ca dropout-ul sa fie 0.005
    super().__init__()
    self.n = n
    self.kernel_size = kernel_size
    self.channels_conv = channels_conv
    self.padding = padding
    self.features = features
    #fiindca am facut automatizat aceste convolutii, trebuie sa definesc un modulelist care in limba incepatoriilor ar fi o lista de functii.
    self.layer = nn.ModuleList()
    self.linear = nn.ModuleList()
    self.norm = nn.ModuleList()
    for index in range(self.n):
        if(index == 0): self.layer.append(nn.Conv2d(in_channels=self.channels_conv[index], out_channels=self.channels_conv[index + 1], kernel_size=self.kernel_size))
        else: self.layer.append(nn.Conv2d(in_channels=self.channels_conv[index], out_channels=self.channels_conv[index + 1], kernel_size=self.kernel_size, padding = self.padding))
        self.linear.append(nn.Linear(self.features[index], self.features[index + 1]))
        self.norm.append(nn.BatchNorm2d(self.channels_conv[index + 1], eps=10 ** (-34)))

    self.linear.append(nn.Linear(self.features[self.n], self.n))

def forward(self, output):
    #aplicam progresia liniara
    for i in range(self.n): output = nn.Dropout(0.005)(F.relu(self.norm[i](nn.MaxPool2d(2, 2)(self.layer[i](output)))))
    output = nn.Flatten()(output)
    for i in range(self.n + 1): output = nn.Dropout(0.005)(self.linear[i](output))
    return nn.Softmax(dim = 1)(output)
```

Maximul acuratetiilor scoase local este de 0.737777, iar pe kaggle 0.68581, pentru 10 epoci fiind:

The training is started... Please Wait
The loss for epoch 1: 0.7406267523765564 with accuracy 68.53333333333333
The loss for epoch 2: 0.6852779388427734 with accuracy 65.04444444444445
The loss for epoch 3: 0.592341423034668 with accuracy 67.77777777777779
The loss for epoch 4: 0.5610758662223816 with accuracy 67.64444444444445
The loss for epoch 5: 0.5653867125511169 with accuracy 68.46666666666667
The loss for epoch 6: 0.5531539916992188 with accuracy 72.64444444444445
The loss for epoch 7: 0.5680323243141174 with accuracy 72.84444444444445
The loss for epoch 8: 0.5651589035987854 with accuracy 69.53333333333333
The loss for epoch 9: 0.5521035194396973 with accuracy 72.04444444444445
The loss for epoch 10: 0.5550351738929749 with accuracy 69.88888888888889

2. Support vector machine

Modelul este preluat din pachetul "sklearn" şi singura chestie ce-as putea face e sa doar citesc si sa bag datele in respectiva clasă.
Aproximarea are acuratetia de 0.45

Observaţie. *Acest model nu a fost incarcat pe Kaggle din motivul acurateţei mici*

Cap 4: Concluzii
O concluzie este că nu s-a meritat sa folsesc SVM-ul, din cauza acurateţii mici şi consider că e mult mai bun CNN-ul. In ciuda lipsei de timp, nu am introdus matricea de confuzie.