Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Umelá inteligencia

2023/24

Zadanie 3 – Klasifikácia datasetov scikit-learn

Lukáš Lovás
7.12.2023

Contents

Lukáš Lovás
7.12.2023

Úloha

Vaša úloha zahŕňa nasledujúce kroky:

1. Zvoľte si a načítajte dataset zo scikit-learn a preskúmajte jeho štruktúru a obsah.
2. Definujte architektúru neurónovej siete, ktorá bude vhodná pre klasifikáciu na základe zvoleného datasetu.
3. Vhodne rozdeľte dataset na trénovaciu a testovaciu množinu.
4. Trénujte zodpovedajúcu neurónovú sieť na trénovacej množine dát a sledujte jej schopnosť naučiť sa vzory a vzťahy medzi triedami a atribútmi.
5. Vyhodnoťte výkonnosť neurónovej siete pomocou relevantných metrík (aspoň tri) a grafických vizualizácií.
6. Vyskúšajte viacero typov neurónových sietí (tiež aspoň tri) a diskutujte o ich plusoch a mínusoch.

Dataset

Zvolený dataset na toto zadanie bol dataset IRIS, ktorý obsahuje údaje o troch rôznych druhoch kvetov kosatcov (Iris setosa, Iris versicolor a Iris virginica).

Každý druh kvetu je reprezentovaný 50 vzorkami, takže celkový počet vzoriek v datasete je 150. Pre každú vzorku sú zaznamenané nasledujúce atribúty:

Dĺžka okvetných lístkov (sepal length): Dĺžka okvetných lístkov kvetu.

Šírka okvetných lístkov (sepal width): Šírka okvetných lístkov kvetu.

Dĺžka okvietia (petal length): Dĺžka okvietia kvetu.

Šírka okvietia (petal width): Šírka okvietia kvetu.

Tieto atribúty sa merajú v centimetroch. Cieľom je klasifikovať kvety do jedného z troch druhov na základe týchto parametrov.

Lukáš Lovás
7.12.2023

Použité frameworky a knižnice

Zadanie bolo spracované v programovacom jazyku python, s využitím frameworku Pytorch určeným na prácu s neurónovými sietami a klasifikačnými modelmi. Ďalšie použité knižnice sú:

datetime – určenie času na vytvorenie adresára

os – vytvorenie adresára

numpy – numpy polia (listy) a určenie unikátnych hodnôt

pandas – práca s Dataframe

seaborn – vizualizácia confusion matrix

sklearn.datasets – načítanie datasetu

from sklearn.preprocessing – škálovanie dát

from sklearn.model_selection – rozdelenie datasetu na trénovaciu a testovaciu množinu

from sklearn.metrics – funkcie na confusion matrix a určenie presnosti výsledku

import tqdm – progress bar pri prebiehaní trénovania

import matplotlib.pyplot – grafy

Lukáš Lovás
7.12.2023

Architektúra neurónových sietí

V tomto zadaní boli implementované tri rôzne architektúry neurónových sietí na klasifikáciu datasetu Iris:

Feedforward Neural Network s ReLU aktivačnou funkciou

Tento model má vstupnú vrstvu s lineárnym prepojením 4 na 50 neurónov, podľa vstupných argumentov datasetu. Nasleduje skrytá vrstva (50 a 50 neurónov). Výstupná vrstva má prepojenie 50 na 3 podľa počtu klasifikačných tried datasetu. Na dátach je počas spracovávania využitá ReLU aktivačná funkcia dvakrát.

Feedforward Neural Network s kombinovanými aktivačnými funkciami

Tento model má vstupnú vrstvu s lineárnym prepojením 4 na 64 neurónov, podľa vstupných argumentov datasetu. Nasleduje skrytá vrstva (64 a 32 neurónov). Ďalšia skrytá vrstva má prepojenie 32 na 16, a ako posledná, výstupná vrstva, ktorá má prepojenie 16 na 3 neuróny, podľa počtu klasifikačných tried datasetu. Na dátach je počas spracovávania využitá ReLU aktivačná funkcia dvakrát, jedenkrát softmax aktivačná funkcia na výstupnej vrstve, a taktiež je upravený počet neurónov. Táto kombinácia aktivátorov a neurónov je užitočná pre úlohy klasifikácie.

Recurrent Neural Network

Tento model využíva 64 skrytých neurónov a výstupnú vrstvu s 3 neurónmi (počet tried v datasete). Tento model využíva rekurentné spojenia medzi časovými krokmi, čo znamená, že berie do úvahy aj minulé kroky, ktoré spravil, a tým zohľadňuje dopad minulých rozhodnutí na ďalšie rozhodnutia.

Lukáš Lovás
7.12.2023

Rozdelenie datasetu na sadu trénovacích a testovacích dát

Rozdelenie datasetu v tomto zadaní bolo 70/30. Kvôli tomu, že sme pracovali s datasetom s 150 záznamami, čo je pomerne obmedzený počet, bolo vhodné vybrať väčšiu množinu dát na trénovanie, aby sa zaručila čo najväčšia výkonnosť trénovaného modelu.

Trénovanie neurónových sietí

Trénovanie prebiehalo pomocou jednoduchej implementácie trénovacieho algoritmu, ktorý posunie dáta modelu na trénovanie, vypočíta pomer správnych určení výsledných dát oproti reálnym výsledkom, pomocou nich určí tzv. „gradients“, to sú vektory, ktoré ukazujú smer a veľkosť, ako majú byť aktualizované parametre modelu počas trénovania.

```
def train_feedforward(epochs):
    loss_list = np.zeros(epochs)
    predictions = []

    for epoch in tqdm.trange(epochs):
        optimizer.zero_grad()  # Delete all gradient from each of the
optimized tensors - gradient is a vector that
        # signalizes which parameters are to be updated based on the loss
criterion
        model_output_train = model(x_train)  # Forwarding the training data
        loss_train = loss_criterion(model_output_train, y_train)  # Compare
model output data with real result data
        loss_train.backward()  # Calculate gradients
        optimizer.step()  # Update parameters - neural network learning
        loss_list[epoch] = loss_train.item()

    visualize_training(loss_list)
```

```
def train_rnn(epochs):
    loss_list = np.zeros(epochs)

    for epoch in tqdm.trange(epochs):
        optimizer.zero_grad()
        model_output_train = model(x_train.unsqueeze(1))
        loss_train = loss_criterion(model_output_train, y_train)
        loss_train.backward()
        optimizer.step()
        loss_list[epoch] = loss_train.item()

    visualize_training(loss_list)
```

Lukáš Lovás
7.12.2023

```python
def test_feedforward():
    model.eval()
    with torch.no_grad():
        predictions = []
        for value in x_test:
            value = value.unsqueeze(0)
            y_predicted = model.forward(value)
            predictions.append(y_predicted.argmax().item())

    accuracy = accuracy_score(y_test, predictions)
    print(f'{model.__class__.__name__} Accuracy: {accuracy * 100:.2f}%')
    visualize_testing(predictions)
```

```python
def test_rnn():
    model.eval()
    with torch.no_grad():
        model_output_test = model(x_test.unsqueeze(1))
        _, predictions = torch.max(model_output_test, 1)

    accuracy = accuracy_score(y_test, predictions)
    print(f'{model.__class__.__name__} Accuracy: {accuracy * 100:.2f}%')
    visualize_testing(predictions)
```

Lukáš Lovás
7.12.2023

Štatistiky a vyhodnotenie efektivity neurónových sietí

Všetky štatistiky a celkové vyhodnotenie efektivity z výsledných dát neurónových sietí sa generujú v týchto funkciách:

```
def visualize_training(loss_list):
    # Save file
    output_dir = os.path.join("output", current_time)
    model_name = model.__class__.__name__
    os.makedirs(output_dir, exist_ok=True)
    path = os.path.join(output_dir, model_name)
    os.makedirs(path, exist_ok=True)

    # Loss plot
    plt.figure(figsize=(10, 10))
    plt.xlabel("epoch")
    plt.ylabel("Loss")
    plt.plot(loss_list, label='Losses during training (difference between
computed results and real results)')
    plt.legend()
    plt.savefig(os.path.join(path, 'loss_plot.png'))
    plt.show()
```

```
def visualize_testing(predictions):
    # Save file
    output_dir = os.path.join("output", current_time)
    model_name = model.__class__.__name__
    os.makedirs(output_dir, exist_ok=True)
    path = os.path.join(output_dir, model_name)
    os.makedirs(path, exist_ok=True)

    # Accuracy table
    test_data_df = pd.DataFrame({'Real data': y_test, 'Predicted data':
predictions})
    test_data_df['Correct'] = [1 if corr == pred else 0 for corr, pred in
                               zip(test_data_df['Real data'],
test_data_df['Predicted data'])]
    plt.figure(figsize=(8, 6))
    plt.axis('off')
    tab = plt.table(cellText=test_data_df.values,
colLabels=test_data_df.columns, cellLoc='center', loc='center', )
    tab.auto_set_font_size(False)
    tab.set_fontsize(8)
    tab.scale(0.7, 0.7)
    plt.show()

    # Accuracy plot
    accuracy = accuracy_score(y_test, predictions)
    plt.figure(figsize=(8, 6))
    plt.plot(predictions, label='Indexes of data', marker='o',
linestyle='None')
    plt.plot(y_test, label='Result labels', marker='x', linestyle='None')
    plt.title(f'Test Accuracy: {accuracy * 100:.2f}%')
    plt.xlabel('Indexes')
    plt.ylabel('Classes')
    plt.legend()
```

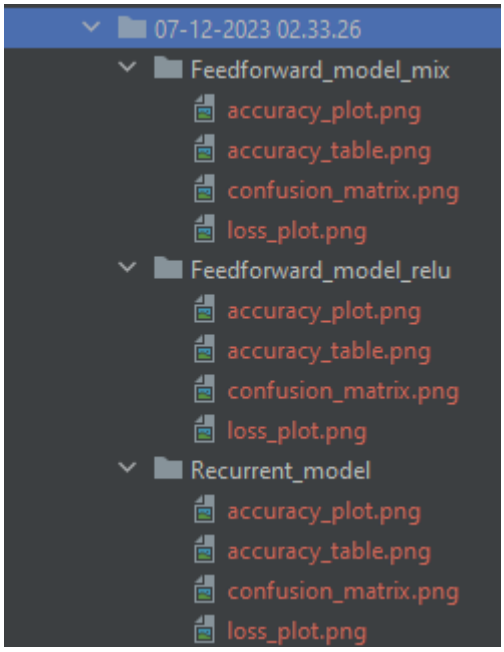
Lukáš Lovás
7.12.2023

```
    plt.savefig(os.path.join(path, 'accuracy_plot.png'))
    plt.show()

    # Confusion matrix
    conf_matrix = confusion_matrix(y_test, predictions)
    plt.figure(figsize=(8, 6))
    sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=np.unique(y_test),
                yticklabels=np.unique(y_test))
    plt.title("Confusion Matrix")
    plt.xlabel("Predicted")
    plt.ylabel("True")
    plt.savefig(os.path.join(path, 'confusion_matrix.png'))
    plt.show()
```

Ešte predtým, než budú vygenerované štatistické grafy, sa dozvieme výslednú presnosť dát spracovaných neurónovou sieťou (výpočet prebieha v test funkciách).

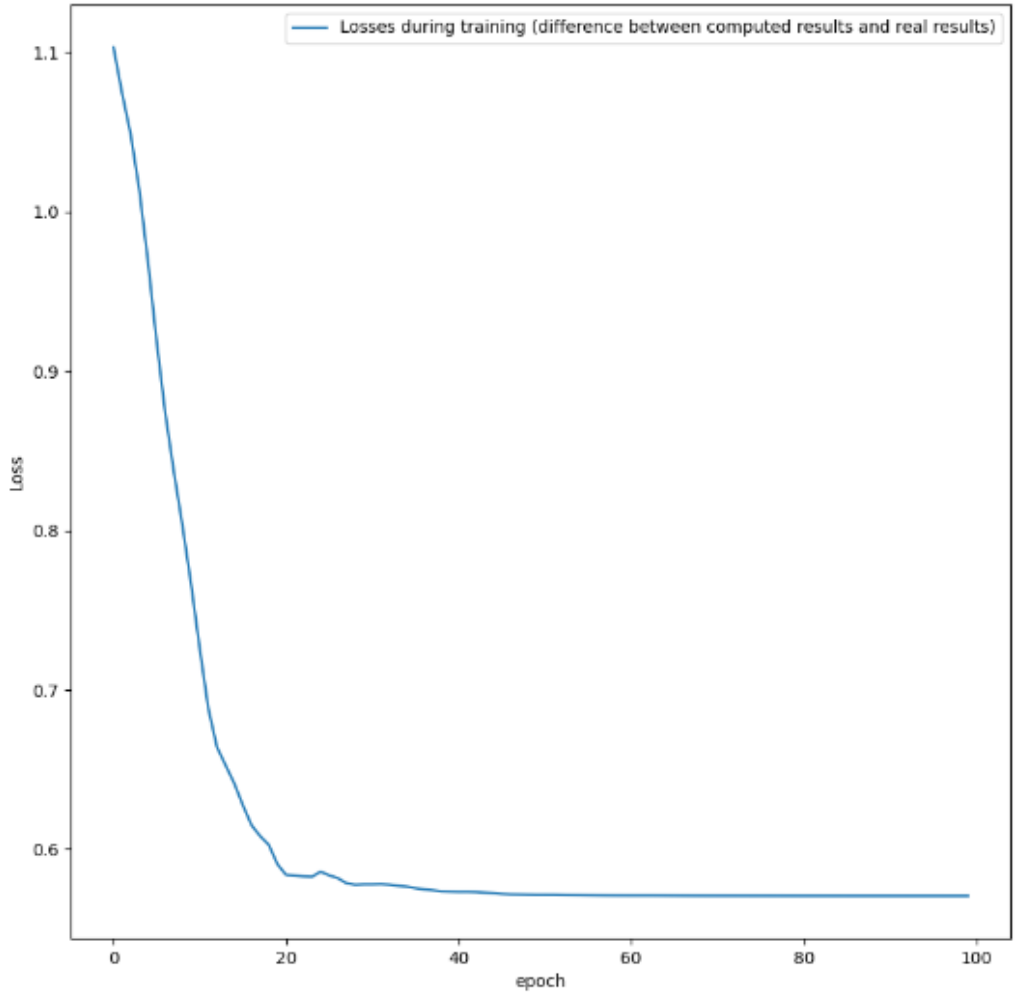
Tieto grafy je vždy možné si pozrieť, pretože sa vygeneruje ich obrázok do danej cesty „output“, kde sú uložené podľa času spustenia programu a neurónovej siete, ktorá poskytla dáta.



Lukáš Lovás
7.12.2023

Typy grafov

Graf strát počas trénovania - nesprávnych klasifikovaní záznamov(loss_plot)



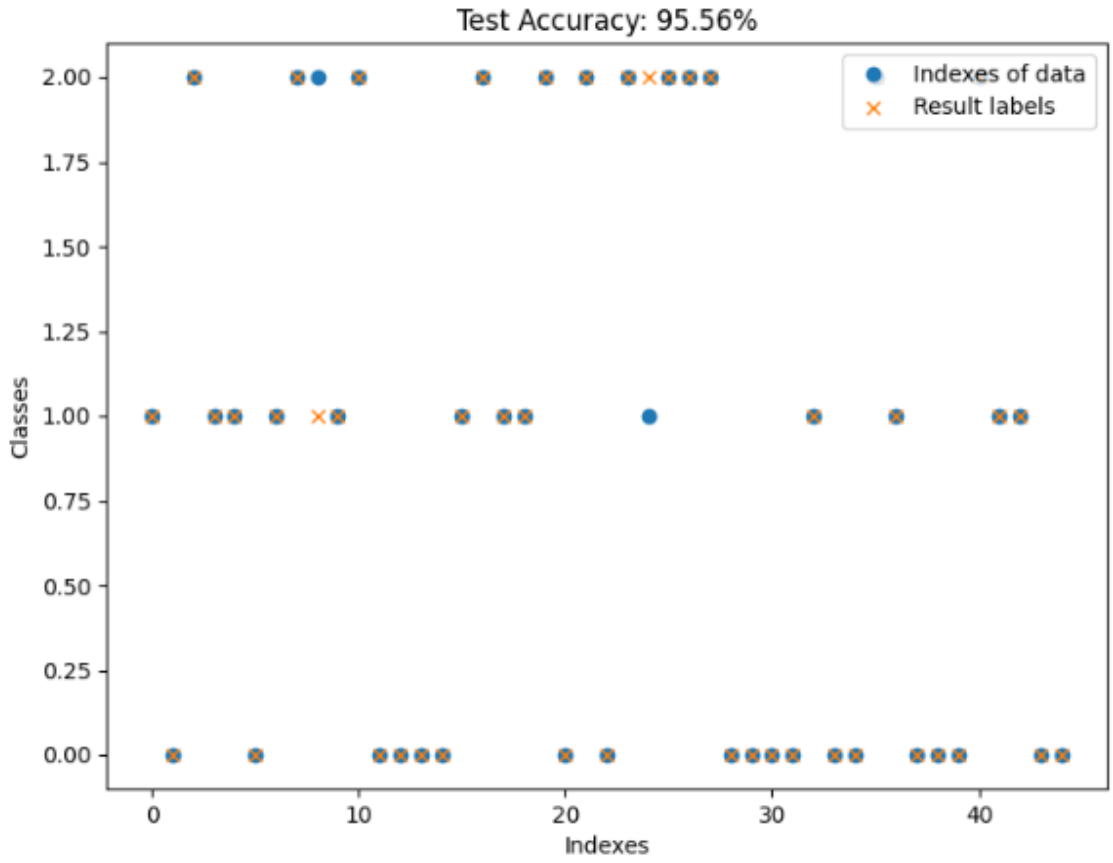
Lukáš Lovás
7.12.2023

Tabuľka klasifikovaní dát neurónovou sieťou a reálnych dát (accuracy_table)

Real data	Predicted data	Correct
1	1	1
0	0	1
2	2	1
1	1	1
1	1	1
0	0	1
1	1	1
2	2	1
1	2	0
1	1	1
2	2	1
0	0	1
0	0	1
0	0	1
0	0	1
1	1	1
2	2	1
1	1	1
1	1	1
2	2	1
0	0	1
2	2	1
0	0	1
2	2	1
2	1	0
2	2	1
2	2	1
2	2	1
0	0	1
0	0	1
0	0	1
0	0	1
1	1	1
0	0	1
0	0	1
2	2	1
1	1	1
0	0	1
0	0	1
0	0	1
2	2	1
1	1	1
1	1	1
0	0	1
0	0	1

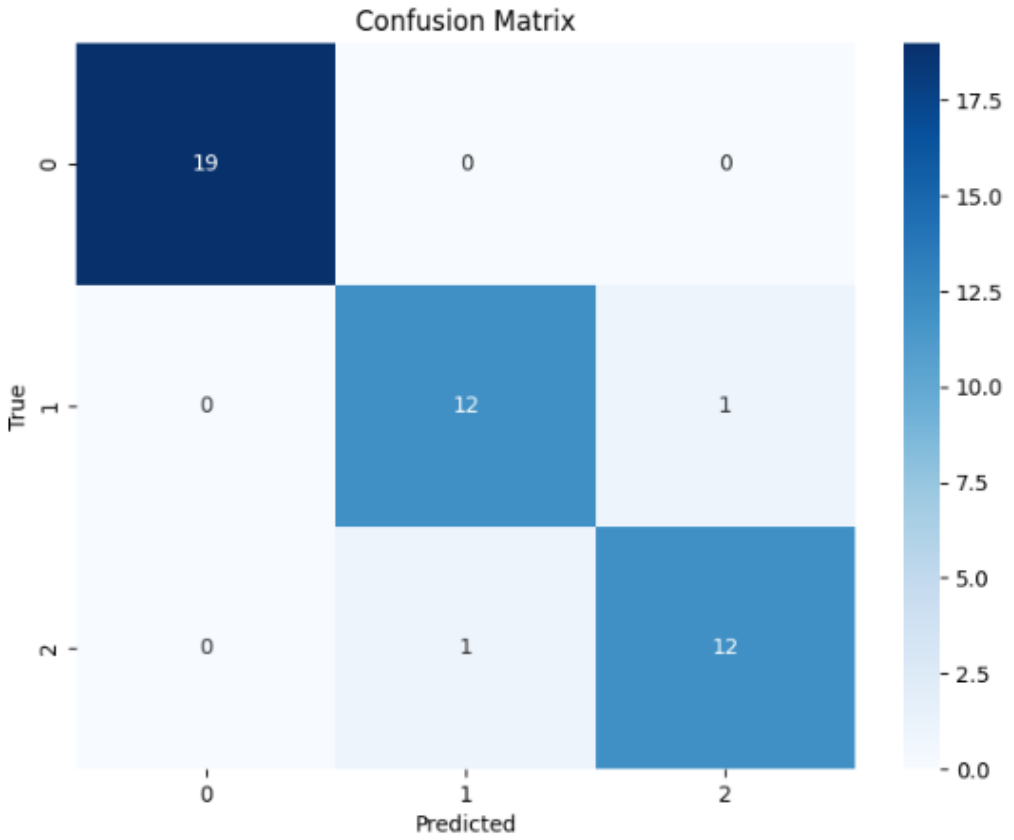
Lukáš Lovás
7.12.2023

Graf presnosti klasifikácie dát (accuracy_plot)



Lukáš Lovás
7.12.2023

Confusion matrix – matica predikovaných a reálnych zaradení dát (confusion_matrix)



Lukáš Lovás
7.12.2023

Plusy a mínusy použitých neurónových sietí

Feedforward Neural Network s ReLU aktivačnou funkciou:

- **Plusy:**
  - Jednoduchá architektúra s jednou skrytou vrstvou, čo môže byť výhodné pre jednoduché úlohy alebo malé datasety.
  - Výhodné použitie ReLU ako aktivačnej funkcie môže pomôcť pri učení sa nelineárnych vzorov.
- **Mínusy:**
  - Pre zložitejšie úlohy alebo datasety môže byť táto architektúra príliš jednoduchá a nedostačujúca.

Feedforward Neural Network s kombinovanými aktivačnými funkciami:

- **Plusy:**
  - Využíva viacero skrytých vrstiev, čo umožňuje modelu pracovať precíznejšie a ulahčuje prácu s tažšími vzormi dát.
  - Použitie softmax funkcie na výstupe je vhodné pre klasifikáciu s viacerými triedami.
- **Mínusy:**
  - Použitie aktivačnej funkcie softmax na výstupe môže spôsobiť problémy pri trénovaní a môže byť citlivý pri práci s „gradientami“.

Recurrent Neural Network

- **Plusy:**
  - Schopnosť zachytávať vzory medzi časovými krokmi môže byť výhodná pre niektoré úlohy.
- **Mínusy:**
  - Môže byť náročnejší na trénovanie a vyžadovať viac dát pre efektívne učenie sa.
  - Citlivý na určovanie zmien parametrov a gradientov
  - Jednoduchá implementácia zložitejšieho algoritmu môže spôsobiť v určitých rozhodnutiach závažné problémy.

Lukáš Lovás
7.12.2023