Laborversuch 3

Versuch	**MLP, ScikitLearn, Keras und PyTorch**
Fach	Int. Syst. u. Masch. Lernen (ISML)
Semester	WS 2023/24
Fachsemester	TIN5/ITS5/WIN5
Labortermine	21.12.2023
Abgabe bis spätestens	12.01.2024

Versuchsteilnehmer

Name: Vorname:

Semester: Matrikelnummer:

Bewertung des Versuches

Aufgabe:	1	2	3	
Punkte maximal:	20	40	40	
Punkte erreicht:				
Gesamtpunktezahl:	/100	Note:		Zeichen:

Anmerkungen:

Aufgabe 1: (4+4+4+4+4 = 20 Punkte)

Thema: Implementierung des Backpropagation-Algorithmus

Betrachten Sie das Python-Modulgerüst `MLP_Backprop.py` zur Implementierung des Backpropagation-Algorithmus:

a) Welche Funktionen enthält das Modul und wozu dienen sie jeweils? Beschreiben Sie jeweils kurz die Funktion (2-3 Sätze).

b) Vervollständigen Sie die Funktion `def forwardPropagateActivity(.)`.

c) Vervollständigen Sie die Funktion `def backPropagateErrors(.)`.

d) Vervollständigen Sie die Funktion `def doLearningStep(.)`.

e) Testen Sie das Modul:

- Erklären Sie kurz was im Hauptprogramm/Modultest genau passiert.
- Erklären Sie kurz die Bedeutung der MLP-Parameter `M`, `eta`, `nEpochs` und `flagBiasUnit`.
- Versuchen Sie das Klassifikationsproblem möglichst schnell (d.h. in wenigen Lernepochen) zu lösen indem Sie gute Werte für die Hyper-Parameter `M` und `eta` finden. Nach wievielen Lernepochen werden bei Ihnen alle Datenpunkte korrekt klassifiziert?

Hinweise:

- Im Skript wurde die Implementierung eines MLP besprochen, für die Vektor-Darstellung siehe (5.55)-(5.63) im Skript. Für Beispiel-Code siehe z.B. auch Kap.5.5, S.135 im alten ILS Skript auf ILIAS.
- Zum Test: Falls zu viele Figuren auf dem Bildschirm dargestellt werden können Sie dies durch Ändern der Zeile `epochs4plot=[-1,0,5,10,50,100,nEpochs-1]` geeignet abändern. Die Liste enthält die Lernepochen für die ein Plot der Entscheidungsgrenze erstellt werden soll.

Aufgabe 2: (40 Punkte)

Thema: Klassifikation von Satellitenbilder Japanischer Wälder mit Scikit-Learn, Keras oder PyTorch

Implementieren Sie mehrere Klassifikatoren für die Bilder Japanischer Wälder aus Versuch 1. Als Klassifikationsmodell verwenden Sie ein MLP und mindestens ein weiteres Verfahren Ihrer Wahl, z.B.:

- KNN
- Lineares Modell (z.B. Least Squares, Logistische Regression, ...)
- Naive Bayes
- Decision Tree
- Random Forest
- Boosting

Zur Implementierung können Sie jeweils eine Machine Learning Bibliothek Ihrer Wahl verwenden, z.B.:

- Scikit-Learn,
- Keras/Tensorflow,
- PyTorch,
- oder Ihr eigenes Klassifikationsmodul von Versuch 1 (evtl. noch das MLP von Aufg.1 darin integrieren!)

Vergleichen Sie jeweils die Klassifikationsleistung (Kreuzvalidierung für $S = 3$) sowie die benötigte Zeit für den Lernvorgang (auch mit Ihrem Ergebnis von Versuch 1). Mit welchem Modell und welchen Hyperparametern erreichen Sie die besten Ergebnisse?

Hinweise:

- Sie können die Programmgerüste aus der Vorlesung verwenden (siehe Vorlesungsfoliensatz "Python Machine Learning" zu Scikit-Learn und Keras)
- Siehe auch die jeweiligen Online-Dokus (mit Beispiel-Code) zu den Machine-Learning-Bibliotheken.
- Für Einlesen und Aufbereiten der Daten können Sie wie bei Versuch 1 vorgehen (siehe dortige Programmgerüste).
- Versuchen Sie die Hyperparameter des jeweiligen Modells zu optimieren (beim MLP z.B. Anzahl Layers, Anzahl Neurone pro Layer, Lernrate, Annealing, Optimizer, Gewichtsinitialisierung, etc.). Welche Hyperparameter zur Verfügung stehen erfahren Sie auf der Online-Doku zu den jeweiligen Modell-Konstruktoren.
- Hinweise für MLP mit Scikit-Learn: Klassifikationsmodell ist `MLPClassifier`, siehe Online-Doku. Am besten erstellen Sie eine `Pipeline` aus `StandardScaler` und `MLPClassifier`. Für das Trainieren bzw. Kreuzvalidieren verwenden Sie am bestene eine "stratified" Kreuzvalidierung, Klasse `RepeatedStratifiedKFold`, z.B. mittels `cv = RepeatedStratifiedKFold(n_splits=3, n_repeats=3, random_state=1)`. Im Gegensatz zur "normalen" Kreuzvalidierung wählt die "stratified" Variante repräsentative Datensets aus, bei denen etwa die Klassenverteilung ähnlich ist wie bei im Geseamtdatenset. Das eigentliche Training und die Validierung können Sie dann etwa mittels `n_scores = cross_val_score(pipeline, X, T, scoring='accuracy', cv=cv, n_jobs=-1, error_score='raise')` erledigen.

- Hinweise für MLP mit Keras: Verwenden Sie die Klassen `keras.Sequential()` und `keras.layers.Dense()`. Falls Sie als Optimierer stochastischen Gradientenabstieg wählen (etwa mittels `optimizer=tf.keras.optimizers.SGD(learning_rate=lr_schedule, momentum=0.0, nesterov=False, name='SGD')`), dann empfiehlt es sich für das "Annealing" einen Lernraten-Scheduler zu benutzen (etwa mittels `lr_schedule = keras.optimizers.schedules.ExponentialDecay(initial_learning_rate=...,decay_steps=..., decay_rate=...)`). Kompilieren des Modells nicht vergessen (etwa mittels `mlp_model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])`). Sie brauchen keine volle Kreuzvalidierung zu machen. Es reicht wenn Sie die Gesamtdaten zunächst in Trainings- und Testdaten splitten (z.B. mittels X_train, X_test, T_train, T_test = sklearn.model_selection.train_test_split(X, T, test_size=0.3). Verwenden Sie dann das Trainings-Set zum Trainieren und Validieren (etwa mittels `mlp_model.fit(X_train, T_train, batch_size=batch_size, epochs=maxEpochs, validation_split=0.2)`). Und das Test-Set zum abschließenden Test (etwa mittels `score = mlp_model.evaluate(X_test, T_test, verbose=0)`).

Aufgabe 3: (40 Punkte)

Thema: Regression der Airfoil-Noise-Daten mit Scikit-Learn, Keras oder PyTorch

Implementieren Sie mehrere Regressionsmodelle für die Airfoil-Noise-Daten von Versuch 2. Als Regressionsmodell verwenden Sie ein MLP und mindestens ein weiteres Verfahren Ihrer Wahl, z.B.:

- KNN
- Naive Bayes
- Lineares Modell (z.B. Least Squares)
- Random Forest
- Boosting

Zur Implementierung können Sie jeweils eine Machine Learning Bibliothek Ihrer Wahl verwenden, z.B.:

- Scikit-Learn,
- Keras/Tensorflow,
- PyTorch,
- oder Ihr eigenes Regressionsmodul von Versuch 2 (evtl. noch das MLP von Aufg.1 darin integrieren!)

Vergleichen Sie jeweils die Klassifikationsleistung (Kreuzvalidierung für $S = 3$) sowie die benötigte Zeit für den Lernvorgang (auch mit Ihrem Ergebnis von Versuch 1).

Hinweise: Für Einlesen und Aufbereiten der Daten können Sie wie bei Versuch 1 vorgehen (siehe dortige Programmgerüste). Ansonsten siehe Hinweise zu Aufg.2.