Laborversuch 3

Versuch	**Neuronale Netze und Backpropagation**
Fach	Intelligente Lernende Systeme
Semester	WS 2021/22
Fachsemester	CPS 5 und ITS5/WIN5
Labortermine	16.12.2021
	23.12.2021
Abgabe bis spätestens	14.01.2022

Versuchsteilnehmer

Name: Vorname:

Semester: Matrikelnummer:

Bewertung des Versuches

Aufgabe:	1	2	3	4	5
Punkte maximal:	10	20	15	15	20
Punkte erreicht:					

Gesamtpunktezahl: /80 Note: Zeichen:

Anmerkungen:

Aufgabe 1: (4+2+2+2 = 10 Punkte)

Thema: Lineare Separierbarkeit, Perzeptron versus Multi-Layer-Perceptron (MLP)

Gegeben seien die folgenden Trainingsdaten: Zu Klasse 1 gehören die Datenpunkte $\{(-2,-1)$, $(-2,2)$, $(-1.5,1)$, $(0,2)$, $(2,1)$, $(3,0)$, $(4,-1)$, $(4,2)\}$ und zu Klasse 2 die Datenpunkte $\{(-1,-2)$, $(-0.5,-1)$, $(0,0.5)$, $(0.5,-2)$, $(1,0.5)$, $(2,-1)$, $(3,-2)\}$.

a) Zeichnen Sie diese Datenpunkte in ein (2D) Koordinatensystem.

b) Ist diese Datenmenge linear separierbar?

c) Kann ein einschichtiges Perzeptron diese Datenmenge fehlerfrei klassifizieren?

d) Kann ein MLP diese Datenmenge fehlerfrei klassifizieren?

Aufgabe 2: (4+4+4+4+4 = 20 Punkte)

Thema: Implementierung des Backpropagation-Algorithmus

Betrachten Sie das Python-Modulgerüst `V3A2_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 bereits besprochen, inklusive Beispiel-Code (siehe Kap.5.5, S.135 im alten Skript bzw. S.137 im neuen Skript; oder SB6, Kontrollaufg.6.1, Musterlösung ab S.89).
- 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 3: (2+4+3+6 = 15 Punkte)

Thema: Integration des Backpropagation-Algorithmus in einem Modul für Klassifikation

Betrachten Sie das Python-Modul `V3A3_MLP3_Classifier.py` zur Integration des Backpropagation-Algorithmus mit dem Klassifikations-Modul `V1A2_Classifier.py` von Versuch 1 und versuchen Sie es zu verstehen:

a) Warum ist es sinnvoll den Klassifikations-Algorithmus von Aufgabe 2 in das Klassifikationsmodul von Versuch 1 zu integrieren?

b) Welche Methoden enthält die Klasse `V3A3_MLP3_Classifier` und welchem Zweck dienen sie?

c) Was für Probleme ergeben sich beim Lernen mit dem Backpropagation-Algorithmus bei einer festen Lernrate η? Wie wird dieses Problem hier gelöst? Welche Parameter sind hierfür relevant und welche Rolle haben sie genau?

d) Testen Sie das Modul:

- Erklären Sie kurz was im Hauptprogramm/Modultest genau passiert.
- Erklären Sie kurz die Bedeutung der Parameter `eta0`, `eta_fade`, `maxEpochs`, `nTrials`, und `eps`.
- Versuchen Sie die beiden Klassifikationsproblem möglichst schnell (d.h. in wenigen Lernepochen) und mit einem möglichst kleinen Netz (wenige Hidden Units) zu lösen indem Sie gute Werte für die Hyper-Parameter `M` und `eta0`, `eta_fade` und `maxEpochs` finden. Wie wird der neue Datenpunkt (blauer Kreis) klassifiziert? Dokumentieren Sie Ihre Ergebnisse.

Hinweise:

- Das hier verwendete numerische Verfahren zur Schrittweitensteuerung beim Lernen nennt man auch **Simulated Annealing** (engl. für Simuliertes Abkühlen). Hierbei sucht man die Lösung zuerst mit grober Auflösung (d.h. großer Schrittweite bzw. "hoher Temperatur") um schnell in gute Parameterbereiche zu kommen. Im weiteren Verlauf des Lernens sucht man mit immer feinerer Auflösung (d.h. kleinerer Schrittweite bzw. "niedrigerer Temperatur") um das Optimum möglichst gut zu treffen.

Aufgabe 4: (3+5+7 = 15 Punkte)

Thema: Klassifikation von Satellitenbilder Japanischer Wälder mit einem MLP

Sie sollen mit Hilfe der MLP-Klasse von Aufgabe 3 wieder verschiedene japanischer Wald-Typen auf Satellitenbildern klassifizieren (vgl. Versuch 1, Aufgabe 4).
Hierfür enthält die Datensammlung `ForestTypesData.csv` (siehe Praktikumsverzeichnis) $N = 524$ Datensätze, wobei jeder Datensatz aus $D = 27$ Bild-Merkmalen der Satellitenaufnahmen besteht (deren genaue Bedeutung uns hier nicht näher zu interessieren braucht). Zusätzlich enthält jeder Datensatz in der ersten Spalte ein Klassenlabel, welches den Datensatz einer von $K = 4$ Klassen zuordnet. Hierbei bedeuten: 's'=Sugi-Zypressenwald, 'h'=Hinoki-Zypressenwald, 'd'=Mischlaubwald, 'o'=unbewaldet.

a) Versuchen Sie zunächst den Aufbau des Programmgerüsts `V3A4_MLP_ForestClassification.py` zu verstehen. Vergleiche Versuch 1, Aufgabe 4.

b) Wozu braucht man hier die Klasse `DataScaler` aus dem Modul `V2A2_regression`?

c) Finden Sie möglichst gute Hyper-Parameter `M`, `lmbda`, `flagBiasUnits`, `eta0`, `eta_fade`, `maxEpochs`, `flagScaleData` um bei einer $S = 3$-fachen Kreuzvalidierung eine möglichst gute Accuracy und Verwechslungsmatrix zu erhalten.

Hinweise:

- Sie können das Programmgerüst `V3A4_MLP_ForestClassification.py` aus dem Praktikumsverzeichnis verwenden.

Aufgabe 5: (10+10 = 20 Punkte)

Thema: Implementierung eines neuronalen Netzwerkes mit Scikit-Learn und Keras

Implementieren Sie das Neuronale Netz der vorigen Aufgabe 4 in

a) Scikit-Learn

b) Keras/Tensorflow

Vergleichen Sie jeweils die Klassifikationsleistung mit der vorigen Aufgabe.

Hinweise:

- Sie können die Programmgerüste aus der Vorlesung verwenden (siehe Vorlesungsfoliensatz "Python Machine Learning" zu Scikit-Learn und Keras).
- Für Einlesen und Aufbereiten der Daten können Sie `V3A4_MLP_ForestClassification.py` wiederverwenden.
- Hinweise für 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 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)`).