Microsynapse

fejlesztői dokumentáció

Microsynapse

A Microsynapse C-könyvtár alkalmas egyszerű gépi tanulási feladatokat gyors futási idővel elvégezni. Kézhez kapott formájában bánni tud kevés adatot és kevés rejtett réteget igénylő folyamatok modellezésére, azonban a primitív hibafüggvény és a robusztus optimalizáló algoritmusok hiánya miatt a könyvtár a fejlesztő által 'kiegészítésre' szorul, ha komplexebb problémákat szeretne vele megoldani. A könyvtár használatának további előnye, hogy a standard C-könyvtárakon kívül semmilyen harmadik féltől származó modult sem igényel.

Matrix

A matrix.h egyetlen származtatott típusa a dim, a mátrixok (több dimenziós, dinamikus tömbök) sorainak és oszlopainak számának tárolására használjuk, a dataframe.h-ban szinte minden függvényhez szüksége van rá a programnak.

A find_max() függvény egy 2D-s dinamikus tömböt és annak dimenzióját veszi át, valamint azt, hogy előjelesek típusúak-e a számok. Mindkét esetben végigfuttat egy ciklust a mátrixon, és megtalálja a legnagyobb abszolútértékű elemet.

Az s_scale() függvény paraméterül egy 2D-s tömböt, annak dimenzióját, valamint a tömb abszolút maximumát kapja, és az utóbbihoz skáláz minden értéket, így a mátrix összes eleme -1 és 1 közti lesz.

A transpose() függvény transzponálja a paraméterként adott 2D-s tömböt(oszlopaiból sort, soraiból pedig oszlopot képez). Felszabadítani a free_transpose függvénnyel lehet, ha az eredeti mátrix dimenzióit akarjuk használni a továbbiakban.

A dinamikus 2D-s tömböket a free_matrix() függvénnyel szabadíthatja fel a felhasználó, paraméterül magát a tömböt, és annak dimenzióit kapja meg.

Dataframe

A dataframe modul alapvető építőeleme a dataframe típus. Ez egy 2D-s tömböt, valamint annak dimenzióit tárolja. Ez nagyban meg fogja könnyíteni a felhasználó dolgát számos területen, mint például a fájlbeolvasásnál, vagy a háló tanításakor.

A get_dim() függvény egy fájlt és a tokeneket elválasztó karaktert kapja paraméterül, és dim típusban adja vissza a fájl 'mértetét'.

A min() csupán segédfüggvény, a head() használja.

A read_csv() függvény az egyik legkulcsfontosságúbb függvénye a könyvtárnak. Egy adatokat tartalmazó fájl elérési útvonalát, és a tokeneket elválasztó karaktert veszi át, majd megnyitja a fájlt, és 2D-s tömbbe írja annak tartalmát (valós számokat).

A head() függvény abban nyújt segítséget a felhasználónak, hogy megmutatja, hogyan néz ki az adott dataframe. Az első öt (vagy ha ennél kevesebb sor van, akkor az összes) sort kiírja.

A free_df() tulajdonképpen csak egy átcímkézett free_matrix() függvény.

Net

Ebben a modulban található a könyvtár velős része. A neurális hálók létrehozásáért, tanításáért, teszteléséért, felszabadításáért felelős.

typedef enum activation: aktivációs függvények felsorolt típusa. Lehet: *SIGMOID, RELU, TANH*.

split típus: a train-test adatok felosztásában játszik szerepet. Elemei egy train, és egy test dataframe.

neuron típus: egy rétegen belüli neuronok ebből a típusból képződnek

output, output_delta: a neuron kimenete (aktivált összege), valamint annak deriváltja

z, z_delta: bemenetek és az eltolássúly összege, ennek deriváltja

bias, bias_delta: eltolássúly, annak deriváltja

weights, weight_deltas: a súlyokra mutató tömb, a súlyok deriváltjaira mutató tömb

layer típus: a modellen belüli rétegek belőle képződnek.

no_neurons: a rétegen belül található neuronok száma

activation: aktivációs függvény

neuron* neurons: a neuronokra mutató tömb

model típus: a neurális háló alapvető paramétereit tárolja, annak összes eleme ezen keresztül lesz elérhető a felhasználó számára.

no_inputs: a háló bemeneteinek száma

no_outputs: a háló kimeneteinek száma

no_hidden_layers: a hálóban található rejtett rétegek száma

added_layers: az add() függvénnyel hozzáadott rétegek száma

struct layer* layers: a modell összes rétegét tároló dinamikus tömb.

split train_test split(): a paraméterként megadott dataframe-ből kiválasztja a megfelelő oszlopokat, és a fejlesztő által megadott arányban felosztja azt tanulási és tesztelési célra.

scaled_rand(): 0 és 1 közti, véletlenszerű számmal tér vissza. A súlyok és eltolássúlyok inicializálásában játszik fontos szerepet.

sigmoid(), relu(): aktivációs függvények.

alloc_neurons(): lefoglalja a szükséges helyet egy adott neuron számára.

create_model(): elkészíti a modellt (de nem inicializálja!) a felhasználó által megadott paraméterek szerint.

fwrand(): feltölti a súlyokat és eltolássúlyokat random értékekre.

init(): lefoglalja a modell rétegeinek (és neuronjainak) kellő helyet, és az fwrand()-dal random értékekre inicializálja a súlyokat, valamint az eltolássúlyokat.

add(): a felhasználó ezzel tud hozzáadni réteget a hálóhoz (fontos: annyit és csak annyit, amennyit a create_model()-lel deklarált).

fwd_pass(): betáplálja az inputokat, végigfuttatja azokat a rétegeken, és megadja az outputokat.

out_prop(): a helyes outputok alapján kiszámolja az output rétegre eső hibát. A bwd_pass() használja.

bwd_pass(): minden rétegben kiszámolja a megfelelő súlyok, eltolássúlyok, outputok, és összegek deriváltját.

update_weights(): a bwd_pass() után módosítja a háló belső állapotát a hiba függvényében.

shuffle(): adott nagyságú egészek közül visszaad egyet véletlenszerűen, így a train() függvény nem mindig ugyanabban a sorrendben olvassa be az inputokat és outputokat.

train(): átveszi a modellünket, a bemenet és kimenet dataframe-eket, a tanulási rátát, és az iterációk számát. Visszatérési értéke az utolsó 100 iteráción felhalmozódott hibák összege.

test(): átveszi a modellt, a bemenet és kimenet dataframe-eket, majd minden egyes sorra kiszámolja a modell által vétett hibát.

pred(): a már betanított modellen így tudunk végigfuttatni általunk megadott inputokat. Átveszi a modellt, az inputok tömbjét, és az outputok tömbjét, majd a modell által jósolt eredményeket az outputok tömbjébe illeszti.

free_model(), free_split(): felszabadítják a struktúrákat.

Hibalehetőségek

NULL pointert kap a függvény.

matrix.h	***dataframe.h***	***net.h***	***file.h***
find_max()	get_dim()	train_test_split() (x3)	save_net() (x2)
s_scale()	read_csv() (x2)	create_model()	load_net() (x2)
transpose()	head()	alloc_neurons()	
	free_df()	fwrand()	
		init()	
		feed_input(), fwd_pass()	
		out_prop(), bwd_pass()	
		update_weights()	
		train() (x3)	
		test() (x3)	
		pred() (x3)	

Egyéb hibalehetőségek:

train_test_split: dim sorai és oszlopai nem pozitív egészek, átadott paraméterek mennyisége nem nagyobb, mint nulla

actv: rossz aktivációs függvény megadása

create_model: inputok vagy outputok száma kisebb, mint 1, rejtett réteg NULL

add: a fejlesztő túl sok réteget akar hozzáadni a hálóhoz.

init: a létrehozott rétegek és deklarált rétegek száma nem egyezik meg.

update_weights: tanulási ráta nem nagyobb, mint nulla (valós).

shuffle: a megadott egész kisebb, mint egy

train: iterációk száma nem nagyobb, mint nulla, tanulási ráta nem nagyobb, mint nulla