ΤΕΧΝΗΤΗ *ΝΟΗΜΟΣΥΝΗ – ΥΠΟΛΟΓΙΣΤΙΚΗ ΝΟΗΜΟΣΥΝΗ*

ΚΑΝΟΝΙΚΟΠΟΙΗΣΗ ΔΕΔΟΜΕΝΩΝ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ ΤΜΗΜΑ Π.Μ. ΕΡΓΑΣΤΗΡΙΟ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ Σ.Ε.Π.Μ.

ΛΑΖΑΡΟΣ ΗΛΙΑΔΗΣ ΚΑΘΗΓΗΤΗΣ ΔΠΘ

liliadis@civil.duth.gr

ΚΑΝΟΝΙΚΟΠΟΙΗΣΗ ΔΕΔΟΜΕΝΩΝ – DATA NORMALIZATION

- **ΔΕΝ ΕΙΝΑΙ ΠΡΟΑΠΑΙΤΟΥΜΕΝΟ ΣΤΑ ΤΝΔ! ΟΜΩΣ ΕΞΑΣΦΑΛΙΖΕΙ ΟΤΙ:**
- **ΘΑ ΥΠΑΡΧΟΥΝ ΚΑΙ ΘΕΤΙΚΕΣ ΚΑΙ ΑΡΝΗΤΙΚΕΣ ΤΙΜΕΣ ΣΑΝ ΕΙΣΟΔΟΙ ΣΤΟ ΕΠΟΜΕΝΟ ΕΠΙΠΕΔΟ => ΠΙΟ ΕΥΕΛΙΚΤΗ ΜΑΘΗΣΗ**
- **ΟΤΙ Η ΜΑΘΗΣΗ ΤΟΥ ΤΝΔ ΘΑ ΑΞΙΟΛΟΓΗΣΕΙ ΟΛΕΣ ΤΙΣ ΠΑΡΑΜΕΤΡΟΥΣ ΣΤΟΝ ΙΔΙΟ ΒΑΘΜΟ**
- **ΜΕΤΑΣΧΗΜΑΤΙΖΕΙ ΤΑ ΔΕΔΟΜΕΝΑ ΕΙΣΟΔΟΥ ΣΤΟ ΠΕΔΙΟ ΤΙΜΩΝ ΤΗΣ ΣΙΓΜΟΕΙΔΟΥΣ [0,1] ΣΥΝΑΡΤΗΣΗΣ ΜΕΤΑΦΟΡΑΣ ΚΑΙ ΤΗΣ ΕΦΑΠΤΟΜΕΝΗΣ ΥΠΕΡΒΟΛΟΕΙΔΟΥΣ ΤΑΝΗ [-1,1]**

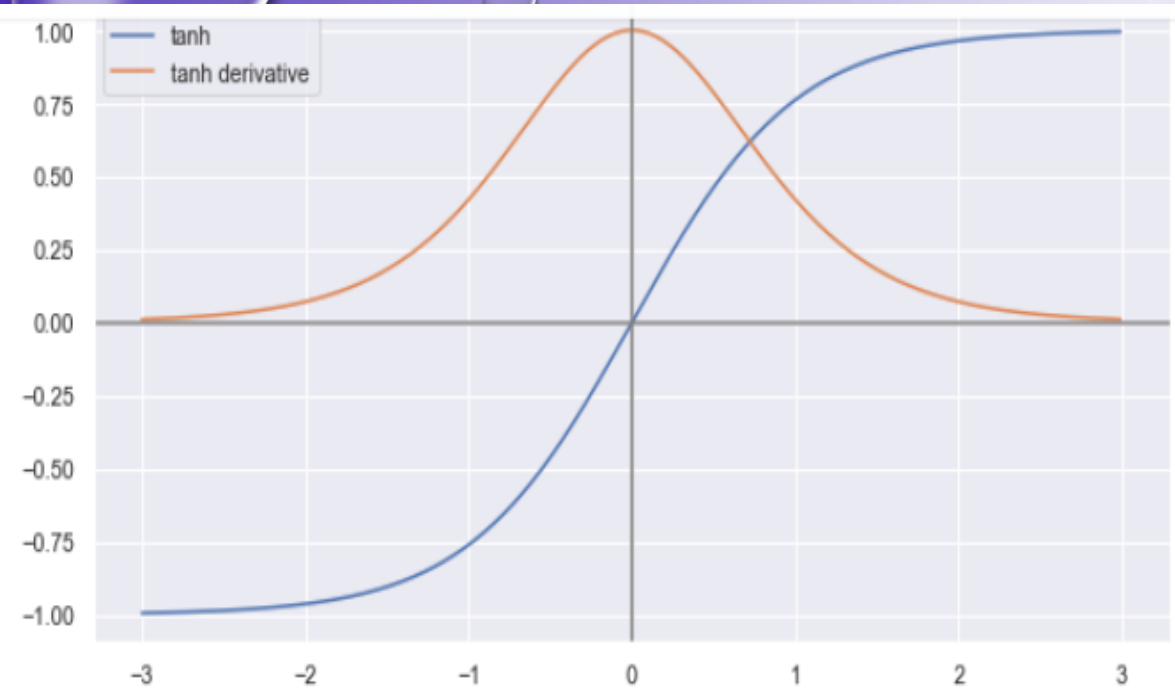
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Σιγμοειδής

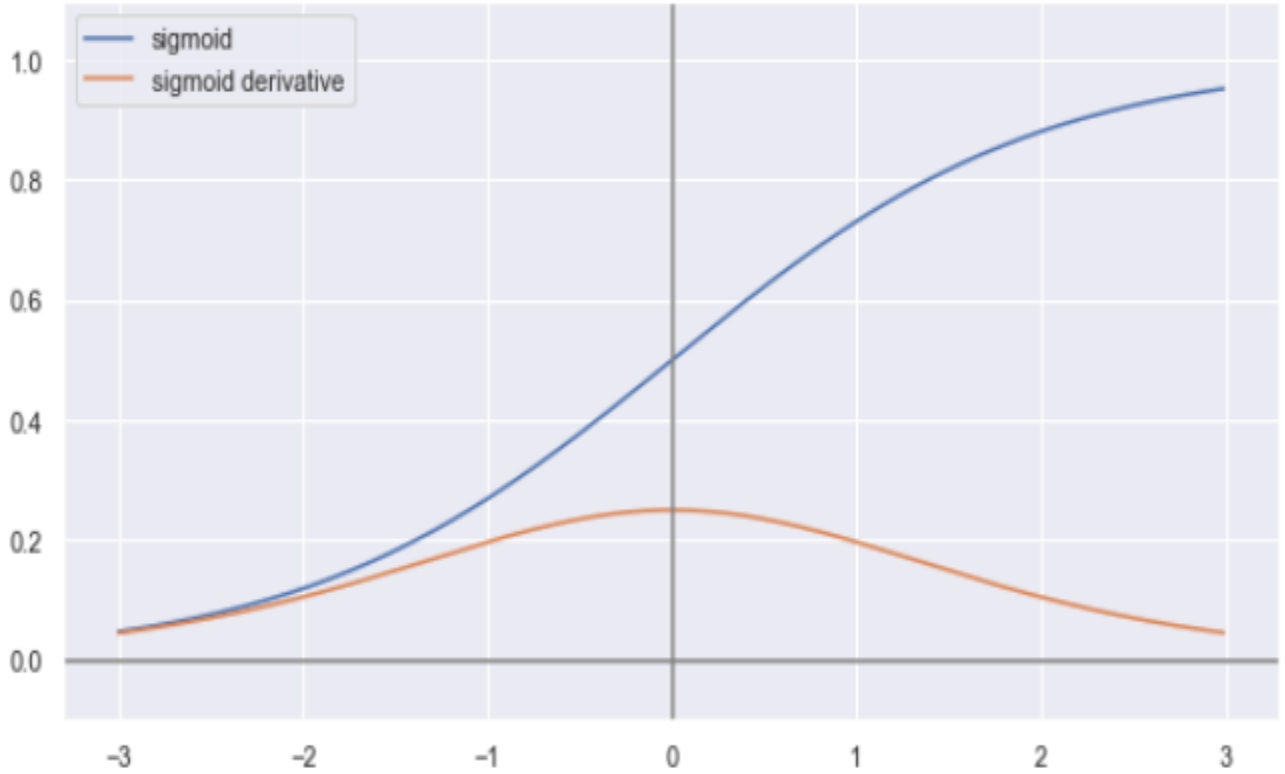
$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Εφαπτόμενη Υπερβολοειδής

ΣΥΝΑΡΤΗΣΕΙΣ ΜΕΤΑΦΟΡΑΣ



Η Εφαπτόμενη Υπερβολοειδής και η Παράγωγός της



ΛΟΓΙΣΤΙΚΗ - ΣΙΓΜΟΕΙΔΗΣ και η ΠΑΡΑΓΩΓΟΣ ΤΗΣ

Το γινόμενο Wi *Xi που μεταβιβάζεται από την είσοδο προς το κρυφό, θα πρέπει να παίρνει μικρές τιμές και σε κάθε περίπτωση οι τιμές των παραμέτρων W1*X1 και W2* X2 να μην έχουν μεγάλη απόκλιση.

Παράδειγμα: έστω ότι έχω 2 παραμέτρους Εισόδου:
όπου η 1η f1 παίρνει τιμές από **0 έως και 0,5** ενώ η 2η f2 από **0 έως 1000**.
Αν η f1 μεταβληθεί κατά 0,5 θα έχει αλλάξει κατά 100% ενώ αν η 2η μεταβληθεί κατά 0,5 θα έχει αλλάξει κατά 0.05% κάτι που δημιουργεί μεροληψία σε βάρος της f1.

Όλοι οι αλγόριθμοι μάθησης εξαρτώνται από αριθμητικές παραμέτρους!!
Αναγκαία όταν έχω μεγάλες διαφορές πχ f1: Ηλικία 0-90 και Εισόδημα 0-εκατομμύρια ή πιο συντηρητικά 0-500.000

Η εμπειρία λέει:

Χρησιμοποιήστε την **TanH** στο κρυφό επίπεδο και την **σιγμοειδή** στο επίπεδο Εξόδου….

Μπορείτε να πειραματιστείτε για να δείτε αν είναι απαραίτητη η κανονικοποίηση στα δεδομένα σας. Αν αποφασίστε να εφαρμόσετε Normalization η κατάλληλη συνάρτηση είναι:

Φυσικά μετά πρέπει να λύσετε ως προς X για να βρείτε τις πραγματικές τιμές στα αποτελέσματα!

$$Z = \frac{x - \mu}{\sigma}$$

Z είναι η κανονικοποιημένη τιμή

x είναι η τρέχουσα τιμή

μ είναι η Μέση Τιμή

σ είναι η Τυπική Απόκλιση

Elevation	Aspect	Slope	Horizontal_D	Vertical_Dist	Horizontal_D	Hillshade_9a	Hillshade_N	Hillshade_3p	Horizontal_Distance_To_Fire_Points
2596	51	3	258	0	510	221	232	148	6279
2590	56	2	212	-6	390	220	235	151	6225
2804	139	9	268	65	3180	234	238	135	6121
2785	155	18	242	118	3090	238	238	122	6211
2595	45	2	153	-1	391	220	234	150	6172
2579	132	6	300	-15	67	230	237	140	6031
2606	45	7	270	5	633	222	225	138	6256
2605	49	4	234	7	573	222	230	144	6228
2617	45	9	240	56	666	223	221	133	6244
2612	59	10	247	11	636	228	219	124	6230
2612	201	4	180	51	735	218	243	161	6222
2886	151	11	371	26	5253	234	240	136	4051
2742	134	22	150	69	3215	248	224	92	6091
2609	214	7	150	46	771	213	247	170	6211
2503	157	4	67	4	674	224	240	151	5600
2495	51	7	42	2	752	224	225	137	5576
2610	259	1	120	-1	607	216	239	161	6096
2517	72	7	85	6	595	228	227	133	5607
2504	0	4	95	5	691	214	232	156	5572

ΠΑΡΑΔΕΙΓΜΑ ΑΠΟΤΕΛΕΣΜΑΤΙΚΟΤΗΤΑΣ ΚΑΝΟΝΙΚΟΠΟΙΗΣΗΣ ΜΕ ΒΑΘΕΙΑ ΜΑΘΗΣΗ

Ενδεικτικά οι πρώτες γραμμές δεδομένων

Χτίζω ένα ΤΝΔ χρησιμοποιώντας Μη Κανονικοποιημένα Δεδομένα

Είναι δεδομένα από το Kaggle που στοχεύουν να κάνουν classification για τον τύπο της δασοκάλυψης - forest cover type

```
'''Using covertype dataset from kaggle to predict forest cover
type'''

#Import pandas, tensorflow and keras

import pandas as pd
from sklearn.cross_validation import train_test_split
import tensorflow as tf
from tensorflow.python.data import Dataset
import keras
from keras.utils import to_categorical
from keras import models
from keras import layers
```

TensorFlow

Βιβλιοθήκη Library Open Source Google Brain Team

Το tf.keras είναι το API της TensorFlow για δημιουργία και εκπ/ση μοντέλων DL.

```
#Import pandas, tensorflow and keras

import pandas as pd
from sklearn.cross_validation import train_test_split
import tensorflow as tf
from tensorflow.python.data import Dataset
import keras
from keras.utils import to_categorical
from keras import models
from keras import layers

#Read the data from csv file

df = pd.read_csv('covtype.csv')

#Select predictors
x = df[df.columns[:54]]

#Target variable

y = df.Cover_Type

#Split data into train and test

x_train, x_test, y_train, y_test = train_test_split(x, y , train_size
= 0.7, random_state =  90)

'''As y variable is multi class categorical variable, hence using
softmax as activation function and sparse-categorical cross entropy
as loss function.'''
```

Η βιβλιοθήκη pandas μετατρέπει δεδομένα csv σε tabular δηλαδή τους δίνει την μορφή πίνακα με στήλες.

Τα δεδομένα διαβάζονται από αρχείο csv

Comma Seperated Vector

```
model = keras.Sequential([
 keras.layers.Dense(64, activation=tf.nn.relu,
 input_shape=(x_train.shape[1],)),
 keras.layers.Dense(64, activation=tf.nn.relu),
 keras.layers.Dense(8, activation=  'softmax')
 ])

model.compile(optimizer=tf.train.AdamOptimizer(),
               loss='sparse_categorical_crossentropy',
               metrics=['accuracy'])


history1 = model.fit(
 x_train, y_train,
 epochs= 26, batch_size = 60,
 validation_data = (x_test, y_test))
```

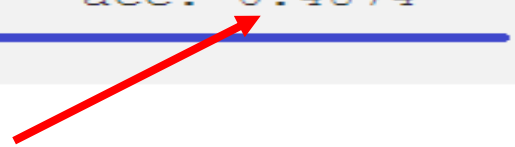
Output:

```
Epoch 1/26 406708/406708 [==============================] – 19s
47us/step – loss: 8.2614 – acc: 0.4874 – val_loss: 8.2531 – val_acc:
0.4880

Epoch 2/26 406708/406708 [==============================] – 18s
45us/step – loss: 8.2614 – acc: 0.4874 – val_loss: 8.2531 – val_acc:
0.4880

…..............

Epoch 26/26 406708/406708 [==============================] – 17s
42us/step – loss: 8.2614 – acc: 0.4874 – val_loss: 8.2531 – val_acc:
0.4880
```



Η Ακρίβεια της ταξινόμησης είναι 48.80%.

```
from sklearn import preprocessing

df = pd.read_csv('covtype.

x = df[df.columns[:55]]
y = df.Cover_Type

x_train, x_test, y_train, y_test = train_test_split(x, y , train_size
= 0.7, random_state = 9

#Select numerical columns which needs to be normalized

train_norm = x_train[x_train.columns[0:10]]
test_norm = x_test[x_test.columns[0:10]]

# Normalize Training Data

std_scale = preprocessing.StandardScaler().fit(train_norm)
x_train_norm = std_scale.transform(train_norm)

#Converting numpy array to dataframe
training_norm_col = pd.DataFrame(x_train_norm,
index=train_norm.index, columns=train_norm.columns)
x_train.update(training_norm_col)
print (x_train.head())

# Normalize Testing Data by using mean and SD of training set

x_test_norm = std_scale.transform(test_norm)
testing_norm_col = pd.DataFrame(x_test_norm, index=test_norm.index,
columns=test_norm.columns)
x_test.update(testing_norm_col)
print (x_train.head())
```

Διαβάζω από το αρχείο csv και το αποθηκεύω σε μορφή πίνακα

Χωρίζω σε Train Validation Testing

Επέλεξε τις 10 στήλες των ανεξάρτητων μεταβλητών για κανονικοποίηση

Κάνε κανονικοποίηση

```
#Build neural network model with normalized data


model = keras.Sequential([
 keras.layers.Dense(64, activation=tf.nn.relu,
 input_shape=(x_train.shape[1],)),
 keras.layers.Dense(64, activation=tf.nn.relu),
 keras.layers.Dense(8, activation=  'softmax')
 ])

model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])


history2 = model.fit(
 x_train, y_train,
 epochs= 26, batch_size = 60,
 validation_data = (x_test, y_test))


#Output:
Train on 464809 samples, validate on 116203 samples
Epoch 1/26 464809/464809 [==============================] - 16s
34us/step - loss: 0.5433 - acc: 0.7675 - val_loss: 0.4701 - val_acc:
0.8022
Epoch 2/26 464809/464809 [==============================] - 16s
34us/step - loss: 0.4436 - acc: 0.8113 - val_loss: 0.4410 - val_acc:
0.8124 Epoch 3/26
.....................
Epoch 26/26 464809/464809 [==============================] - 16s
34us/step - loss: 0.2703 - acc: 0.8907 - val_loss: 0.2773 - val_acc:
0.8893
```

Η ακρίβεια έγινε ίση με: 88.93%