# TensorFlow

## הגדרה

זוהי ספריית קוד פתוח בפייתון שעיקר השימוש שלה הוא ללמידה עמוקה לבנות רשתות נוירונים, אך משמשת גם באופן כללי ללמידת מכונה. לספרייה זו יש שתי גרסאות מרכזיות: גרסה 1 שהיא מורכבת יותר ודורשת יותר הבנה ופירוט שנעשה על ידי המתכנת, וגרסה 2 שהיא פשוטה יותר ומבצעת הרבה דברים מאחורי הקלעים כדי שהקוד יהיה נגיש יותר.

לאחר התקנת TensorFlow, בכל תוכנית צריך לייבא אותו על ידי import. TensorFlow מיובא בדרך כלל תחת הכינוי tf.

import tensorflow as tf

## אובייקטים

ב-TensorFlow כל אובייקט הוא Tensor, כלומר מערך רב ממדי. מערך זה יכול להיות סקלר, וקטור, מטריצה, וקטור מטריצות וכו', ויכול להכיל כל סוג של נתונים. כל אובייקט יכול להיות קבוע או משתנה המייצג ערך או פעולה וצריך לקבל ערך התחלתי.

נהוג לדמות כל אובייקט ב-TensorFlow כאל צומת בגרף. בגרסה 1 לא ניתן לגשת אל הערך או הפעולה השמורה בתוך אובייקט ישירות, אלא יש לפתוח session באמצעות tf.session(), ולאחר מכן, באמצעות שיטת run על ה-session שפתחנו, נכניס את האובייקט ונפעיל את הערך או הפעולה השמורה בו. ניסיון להדפיס אובייקט שלא באמצעות session ידפיס את דרגת המערך ואת סוג המידע השמור באובייקט. ב- TensorFlowגרסה 2 לעומת זאת, אין צורך לפתוח session אלא כשנדפיס את האובייקט הוא יריץ session אוטומטית ויחזיר את הערך שבתוכו.

### קבועים

אלו אובייקטים שנוצרים באמצעות שיטת tf.constant המקבלת מערך רב ממדי כלשהו ומחזירה אובייקט קבוע. לאחר יצירת האובייקט לא ניתן לשנות את ערכו.

>>> a = tf.constant(3)

>>> a

<tf.Tensor 'Const:0' shape=() dtype=int32>

### משתנים

אלו אובייקטים שנוצרים באמצעות שיטת tf.Variable המקבלת מערך רב ממדי כלשהו ומחזירה אובייקט שניתן לשנותו.

>>> var = tf.Variable(3)

>>> var

<tensorflow.python.ops.variables.Variable object at 0x7f41f00abd10>

לאחר שפותחים session יש לאתחל את כל הערכים של כל המשתנים באמצעות הפקודה למטה. בכל פעם שמריצים פקודה זו ערך כל המשתנים חוזר להיות הערך ההתחלתי שקיבלו.

>>> sess.run(tf.global\_variables\_initializer())

### פעולות

כאמור לעיל אובייקט יכול להכיל גם פעולה כלשהי. tf.assign זוהי שיטה המקבלת משתנה והשמה עבורו, ומחזירה את המשתנה לאחר ההשמה. דוגמה לקוד שבו מדפיסים ערך של משתנה בקפיצות קבועות.

>>> import tensorflow as tf

>>> x = tf.Variable(1)

>>> step = tf.constant(2)

>>> update = tf.assign(x, x+step)

>>> sess = tf.Session()

>>> sess.run(tf.global\_variables\_initializer())

>>> for i in range(4):

... print(sess.run(update))

3, 5, 7, 9 # In fact each value will be in a separate row

>>> sess.run(x)

9

>>> sess.run(tf.global\_variables\_initializer()) # Initializes all variables to the initial value

>>> sess.run(x)

1

## שיטות מתמטיות

tf.reduce\_mean - מקבלת וקטור ומחזירה ממוצע של כל הערכים בו.

reduce\_sum – מקבל וקטור וסוכם כל הערכים בו.

tf.abs – מקבל וקטור ומחזיר וקטור של הערכים המוחלטים של כל רכיב בהתאמה.

tf.matmul – מקבל שני וקטורים ומכפיל ביניהם.

tf.zeros – יוצרת טנזור שבו כל הערכים מאותחלים ל-0. מקבלת רשימה של ערכים המציינים את המידדים של הטנזור. ניתן להוסיף גם את סוג הערכים שיכיל.

tf.zeros([3, 4], tf.int32)

tf.nn.sigmoid(tf.matmul(x,W) + b) – יוצרת פונקציית סיגמואיד ל-logistic regression.

## שיטות בלמידת מכונה

tf.train - שיטה שמכינה את האלגוריתם באמצעותו נשתמש כדי לאמן את המודל. ניתן לבחור איזה אלגוריתם להשתמש, בדרך כלל GradientDescentOptimizer המקבל את גודל הצעד , ולאחר מכן האם רוצים מינימום או מקסימום של ה-Loss function.

update = tf.train.GradientDescentOptimizer(0.0001).minimize(loss)

## Linear Regression

דוגמה לשימוש ב-TensorFlow כדי לבצע Linear Regression. המטרה כאן היא למצוא איזה מספר מכפיל את הערכים ב-x כך שמתקבלים הערכים ב-y בהתאמה.

import tensorflow as tf

x = tf.constant([7.01, 3.02, 4.99, 8.])

y\_ = tf.constant([14.01, 6.01, 10., 16.04])

w = tf.Variable(0.) #If we don't use ‘.’ will assume that ‘w’ is an int and fail, because it doesn't cast to float automatically.

y = w\*x

loss = tf.reduce\_mean(tf.pow(y - y\_, 2))

update = tf.train.GradientDescentOptimizer(0.0001).minimize(loss)

sess = tf.Session()

sess.run(tf.global\_variables\_initializer())

for \_ in range(0,1000):

sess.run(update)

print(sess.run(w))

2.00052

## PlaceHolders

זהו אובייקט שבאמצעותו ניתן לשמור מקום בקוד כדי שבהמשך הקוד נוכל להזין את ה-data. משתמשים באובייקטים אלו כאשר רוצים לבנות מודל שיעבוד כל פעם על data אחר. בונים אובייקט מסוג זה באמצעות פונקציה tf.placeholder המקבלת את סוג ה-data, ורשימה עם שני איברים שבה מספר האובייקטים ב-data ומספר התכונות לכל אובייקט. כאשר לא נרצה להגביל אחד מהם נרשום None.

x = tf.placeholder(tf.float32, [None, 1]) # Unlimited data but with only one feature

y\_ = tf.placeholder(tf.float32, [None, 1])

...

sess.run(update, feed\_dict = {x:[[7.01], [3.02], [4.99], [8.]], y\_:[[14.01], [6.01], [10.], [16.04]]})

## Saver

כאשר רוצים במהלך בניית מודל לשמור את המשקולות במצב הנוכחי. שימושים נפוצים ב-Saver:

1. כאשר טוענים data בחלקים נרצה לשמור את המשקולות בין הטעינות.
2. כאשר בניית המודל לוקחת המון זמן, נרצה שכל פרק זמן לשמור מצב קיים מחשש לאיבוד מידע הנובע מהפסקת חשמל וכדומה.
3. כאשר רוצים לבנות מודל חדש אך עדיין לשמור תוצאות של מודל קודם.

saver = tf.train.Saver()

saver.save(sess, filename) # save to file

saver.restore(sess, filename) # load from file

## TensorBoard

אחת האפשרויות היא לראות את הגרף המכיל כל האובייקטים שיצרנו שעליו אנו מריצים את ה-Session.