

TensorFlow

M.B.P.

December 28, 2017

Resumen

Aca va el resumen del trabajo

¿Qué es TensorFlow?

Se trata de una librería de código abierto desarrollada por Google para facilitar el diseño, construcción y entrenamiento de sistemas de aprendizaje profundo.

¿Qué es un tensor?

Un tensor es una generalización de vectores y matrices potencialmente altas dimensiones. Internamente, TensorFlow representa tensores como matrices n-dimensiones de base datatypes.

Un `tf.Tensor` tiene las siguientes propiedades:

- Un tipo de dato (`float32`, `int32`, o cadena, por ejemplo).
- Una forma (es decir, el número de dimensiones que tiene y el tamaño de cada dimensión).

¿Como importar la libreria?

Para importar la libreria de TensorFlow en cualquier proyecto se usa el comando:

```
import tensorflow as tf
```

Tipo de Tensor

Tensorflow trabaja con diferentes tipos de elementos denominados tensor, estos deben ser declarados para su uso dentro de la seccion del tensor.

Tabla de tipos de Tensor

Tipo	Descripcion
tf.float16	Punto flotante de precisión media de 16 bits
tf.float32	Punto flotante de precisión simple de 32 bits
tf.float64	Punto flotante de doble precisión de 64 bits
tf.bfloat16	Punto flotante truncado de 16 bits
tf.complex64	Complejo de precisión simple de 64 bits
tf.complex128	Complejo de doble precisión de 128 bits
tf.int8	Entero con signo de 8 bits
tf.uint8	Entero sin signo de 8 bits
tf.uint16	Entero sin signo de 16 bits

Tabla de tipos de Tensor

Tipo	Descripcion
tf.int16	Entero con signo de 16 bits
tf.int32	Entero con signo de 32 bits
tf.int64	Entero con signo de 64 bits
tf.bool	Booleano
tf.qint8	Entero cuantificado de 8 bits con signo
tf.quint8	Número entero sin signo de 8 bits cuantificado
tf.qint16	Entero cuantificado de 16 bits con signo
tf.quint16	Número entero sin signo de 16 bits cuantificado
tf.qint32	Número entero con signo de 32 bits cuantificado

Declaracion de variables

En la sintaxi de python las variables se declaran de forma sencilla como enteros o flotantes:

```
x = 12    esta variable sera entera.  
y = 2.5   esta variable sera flotante.
```

Cuando se pretende usar TensorFlow la declaracion de las variables cambia teniendo una forma mas especial en la sintaxi:

```
x = tf.constant(3.0, dtype=tf.float32)  
y = tf.constant(4.0)
```

En este caso ambas son reconocidas como tipo float pero declaradas de diferente manera.

Inputs and Readers

Placeholders

TensorFlow proporciona una operación de marcador de posición que debe ser alimentado con los datos sobre la ejecución:

- `tf.placeholder`
- `tf.placeholder_with_default`

Para la alimentación de `SparseTensors` que son de tipo compuesto, hay una función de confort:

- `tf.sparse_placeholder`

Inputs and Readers

Readers

TensorFlow proporciona un conjunto de clases Reader para leer formatos de datos:

- `tf.ReaderBase`
- `tf.TextLineReader`
- `tf.WholeFileReader`
- `tf.IdentityReader`
- `tf.TFRecordReader`
- `tf.FixedLengthRecordReader`

Sintaxis Inputs

Usando placeholder de la siguiente manera:

```
placeholder(dtype, shape=None, name=None)
```

dtype es el tipo de elemento del tensor que sera alimentado.

shape es la forma del tensor para ser alimentado esta puede ser opcional si no se especifica la forma ya que se puede alimentar de cualquier forma.

name nombre para la operacion igual es opcional.

Comandos: operaciones basicas

Comando	Accion
add()	Suma entre dos nodos ($x + y$)
subtract()	Resta entre dos nodos ($x - y$)
multiply()	Multiplicacion entre dos nodos ($x * y$)
div()	Divicion entre dos nodos ($\frac{x}{y}$)
square()	Determina el cuadrado de un nodo (x^2)
pow()	Calcula la potencia de un nodo (x^y)
sqrt()	Determina la raiz cuadrada de un nodo (\sqrt{x})

Sintaxis

Para el uso de cada comando es necesario entender su sintaxis y los elementos que la componen en cada caso están compuestos de tres elementos pero puede variar dependiendo de la función:

Un elemento que se encuentra presente en estos comandos es el **name** el cual da un nombre para la operación a realizar (este es opcional y no afecta al uso de la misma).

Sintaxis

Para la suma con el `add` seria:

```
add(x, y, name=None)
```

- x,y seran un tensor del mismo tipo, los cual pueden ser uno de los siguientes tipos: `half`, `float32`, `float64`, `uint8`, `int8`, `int16`, `int32`, `int64`, `complex64`, `complex128`, `string`

El resultado sera del mismo tipo ocupado en dicha función.

Para la resta usando `subtract` seria:

```
subtract(x, y, name=None)
```

- x,y seran un tensor del mismo tipo, los cual pueden ser uno de los siguientes tipos: `half`, `float32`, `float64`, `uint8`, `int8`, `int16`, `int32`, `int64`, `complex64`, `complex128`

El resultado sera del mismo tipo ocupado en dicha función.

Sintaxis

Para la multiplicación usando `multiply` tenemos:

```
multiply(x, y, name=None)
```

- x,y serán un tensor del mismo tipo, los cual pueden ser uno de los siguientes tipos: `half`, `float32`, `float64`, `uint8`, `int8`, `uint16`, `int16`, `int32`, `int64`, `complex64`, `complex128`

El resultado será del mismo tipo ocupado en dicha función.

Para la división usando `div` tenemos:

```
div(x, y, name=None)
```

- x,y serán un tensor del mismo tipo, el tipo de numerador y denominador deben ser números reales.

El resultado deberá ser el cociente de x,y.

Sintaxis

Para elevar un elemento al cuadrado se usa `square` tenemos:
`square(x, name=None)`

- `x` puede ser un tensor o un `sparsetensor`, el tipo que se puede ocupar es: `half`, `float32`, `float64`, `int32`, `int64`, `complex64`, `complex128`

El resultado debe ser del mismo tipo que `x`.

Para elevar un elemento en cualquier exponente tenemos `pow`:
`pow(x, y, name=None)`

- `x, y` deben ser tensor del mismo tipo, estos tipos son: `float32`, `float64`, `int32`, `int64`, `complex64`, or `complex128`

El resultado debe ser del mismo tipo del tensor.

Sintaxis

Para obtener la raíz cuadrada de un elemento se usa `sqrt`:

`sqrt(x, name=None)`

- `x` es un tensor o un `sparsetensor`, el tipo de este puede ser: `half`, `float32`, `float64`, `complex64`, `complex128`

El resultado debe ser un tensor o un `sparsetensor` del mismo tipo ocupado.

Listing 1: Ejemplo del uso de operaciones

```
import tensorflow as tf

x = tf.placeholder("float")
y = tf.placeholder("float")

suma = tf.add(x, y)

sess = tf.Session()

print ("suma=", sess.run(suma,
                           feed_dict={a: 3, b: 3}))
```

Aqui empieza el capitulo sobre estado