Deep learning con TensorFlow

Gabriel Muñoz Ríos

CCIA - Machine Learning Group

Mayo - 2016

Índice

¿Qué es TensorFlow?

TensorFlow™ es una librería de código abierto para realizar operaciones (numéricas, computación) usando el flujo de datos en un grafo. En este grafo, los nodos representan las operaciones mientras que las aristas representan las estructuras de datos (o tensores).

Originalmente, TensorFlow fue desarrollado por investigadores e ingenieros de *Google Brain Team* y *Google's Machine Intelligence research* para proyectos de machine learning y deep learning. El sistema se ha generalizado tanto que, a día de hoy, se puede aplicar a muchos otros tipos de proyectos.

Algunas características

Algunas de las características que TensorFlow nos ofrece:

- Mucha flexibilidad (Si lo puedes representar como un grafo...)
- Portabilidad real (CPUs, GPUs, Escritorio, Servidor, Móvil...)
- Conexión entre investigación y producción
- Auto diferenciación (TensorFlow se encarga de la derivación por nosotros)
- Opciones de lenguajes (Actualmente: Python y C++. Próximamente: Go, Java, R...)
- Maximizar el rendimiento (Control total de los dispositivos a usar...)

Índice

Conceptos básicos

TensorFlow funciona sobre cinco conceptos principales:

- Representación de operaciones computacionales como un grafo
- Ejecución de dichos grafos en contextos (Sesiones)
- Representación de los datos como tensores
- Mantenimiento del estado del grafo mediante variables
- Facilidad para proporcionar y obtener datos durante la ejecución de operaciones

Grafo y flujo de datos

Un grafo de flujo de datos no es más que un grafo dirigido en el que definimos ciertas operaciones conectadas entre sí.

Normalmente, los nodos del grafo contendrán las operaciones que queremos ejecutar. TensorFlow proporciona un rango amplio de operaciones, desde algunas a bajo nivel como otras que nos abstraen de operaciones más enrevesadas.

Por otro lado, en las aristas definiremos la entrada y salida de las relaciones entre los nodos. Las aristas se encargarán de transmitir de un nodo a otro arrays de datos multidimensionales. Estos arrays de datos será lo que llamaremos tensores, nuestra estructura de datos en TensorFlow.

Tensores compartidos en un flujo de datos en un grafo...Os suena el nombre? :)

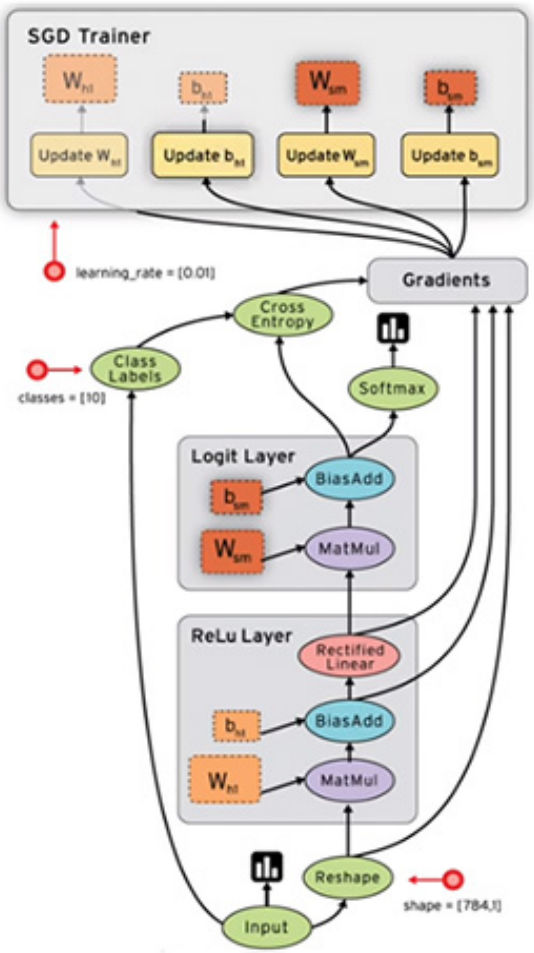
Grafo y flujo de datos

Recapitulando:

Con TensorFlow usamos un grafo para representar operaciones y los datos que usan estas operaciones.

Los nodos del grafo contienen las operaciones a aplicar. Cada operación toma cero o más tensores (aristas), realiza una operación y produce cero o más tensores. Estos tensores son arrays multidimensionales.

Grafo y flujo de datos



El grafo de computación

Cuando escribimos un programa usando TensorFlow, normalmente dividiremos nuestro trabajo en dos fases. La primera será la fase de construcción, donde definimos los nodos del grafo y las conexiones entre ellos. La segunda es la fase de ejecución, donde haciendo uso de un contexto (Sesión) ejecutaremos las operaciones del grafo.

Un ejemplo que se acopla muy bien a esta estructura es una red neuronal. En la fase de construcción crearemos un grafo que represente todas las capas y elementos necesarios de nuestra red. Una vez la tengamos construida, usamos un bucle para ejecutar todas esas operaciones y así comenzar el entrenamiento de la misma.

El grafo de computación

Cada vez que trabajamos con TensorFlow tenemos que usar un grafo donde declarar nuestras variables, operaciones, etc. Por ello, TensorFlow nos ofrece un grafo por defecto que en la mayoría de las ocasiones nos valdrá para ejecutar nuestras operaciones.

Como podéis imaginar, a parte del grafo por defecto, podemos crear grafos, finalizarlos, manipularlos, crear contextos para operaciones, controlar dependencias, etc. Para nuestros ejemplos tendremos suficiente con el grafo por defecto.

El uso de Contexto o Sesión

Ya tenemos un grafo para declarar nuestras operaciones y tensores. Ahora necesitamos una Sesión para ejecutar nuestro grafo.

Podemos ver la Sesión como la parte en la que encapsulamos la ejecución completa de nuestro grafo. Necesitaremos esta Sesión para controlar las operaciones, inicializar, consultar o modificar los valores de los tensores, liberar recursos, etc.

El uso de Contexto o Sesión

El método principal de la Sesión es el método *run()*. Con este método ejecutamos todas las operaciones de nuestro grafo de principio a fin.

Si lo que queremos es evaluar el valor de los tensores, tenemos dos opciones. La primera es usar el método *eval()* y la segunda es indicándole como parámetro al propio método *run()* los tensores de los que queremos obtener el valor durante la ejecución.
Revisaremos este concepto más adelante.

El uso de Contexto o sesión

Dentro del uso de la Sesión podemos diferenciar dos usos principales. En el primero, y el que nos encontramos en la mayoría de casos, podemos ver la Sesión como un contexto *cerrado*, que necesitaremos declarar como la sesión a usar por defecto.

Aparte de esta forma, podemos también hacerlo con lo que se conoce como sesión interactiva. La única diferencia es que este tipo de sesión se auto declara como la sesión por defecto, por lo que no tendremos que declarar la misma para ejecutar el grafo.

Esta forma de trabajar es la que se aconseja en el caso de usar notebooks ya que es más flexible a la hora de trabajar entre distintas celdas.

Tensores

Como ya hemos introducido anteriormente, los programas que implementemos usando TensorFlow usarán como estructuras de datos los tensores.

Todos los datos en TensorFlow se representan usando esta estructura ya que las operaciones que definamos en un grafo de computación solo aceptarán tensores como estructuras de datos para sus operaciones.

Para entender la forma de un tensor, necesitamos conocer las tres propiedades que lo definen: type, rank y shape.

Tensores

- Rank: Es el número de dimensiones del Tensor
- Shape: Es la manera en la que se organizan los datos en el Tensor
- Type: Es el tipo de los datos

Por ejemplo, podríamos tener un tensor de tipo float32 con rank 2 y con shape [3, 4].

Variables

Las variables son las operaciones que mantienen el estado de las ejecuciones de nuestro grafo. Son en realidad buffers en memoria que contienen los valores de nuestros tensores.

En realidad, lo que se hace cuando se declara una variable es declarar una operación que se encarga de inicializarla cuando arranquemos nuestra ejecución y mantener su valor durante la misma.

Fetches y Feeds

TensorFlow nos permite interactuar con nuestros tensores cuando ejecutamos nuestras operaciones (método run en la Sesión).

Para ello, disponemos de dos opciones: obtener el valor de uno o más tensores o modificar el valor de uno o más tensores. Estas operaciones son las que se conocen como *fetch* y *feed*, respectivamente.

Este dinamismo nos permite tener mucho control sobre nuestro grafo de ejecución, permitiéndonos comprobar valores de tensores que nos interesen o ir modificando otros para ver el comportamiento de nuestras operaciones.

Como punto importante, si queremos indicar que una operación va a ser modificada de manera dinámica la definiremos con el tipo *tf.placeholder()*.

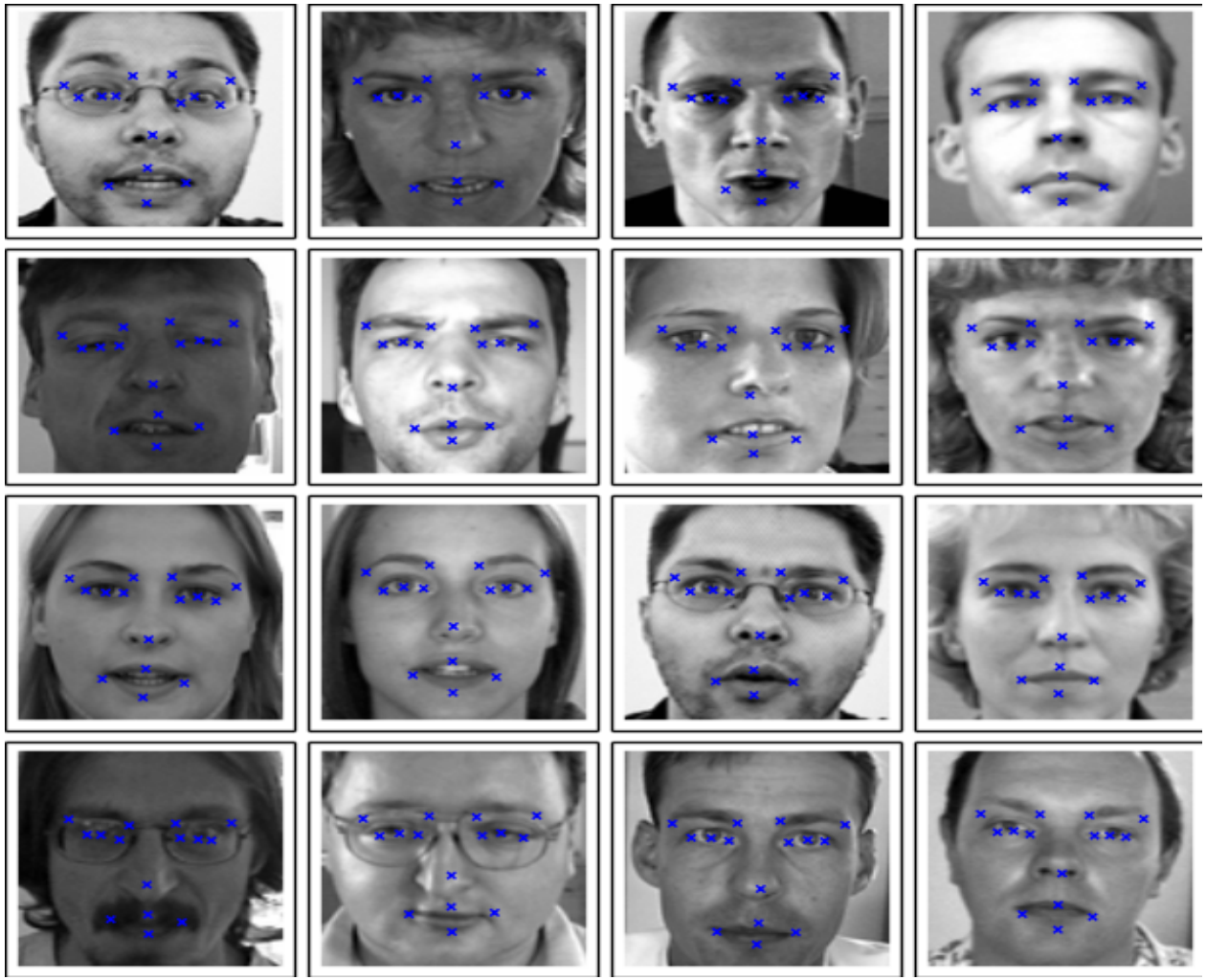
Índice

Dataset

Ya conocemos los conceptos básicos que usa TensorFlow. Es hora de ponernos manos a la obra con nuestros ejemplos. Comencemos con el conjunto de datos.

El conjunto de datos que usaremos es el que tenemos disponible en el reto de Kaggle llamado *Facial Keypoints Detection*.
Básicamente, tendremos que predecir la posición de una serie de puntos en imágenes faciales.

Dataset



Dataset

Los datos vienen disponibles en un archivo CSV y tienen la siguiente forma: (Ver data/training_sample.csv)

Dataset

La selección de este conjunto de datos se debe a varias razones:

- Detectar puntos faciales es un problema complicado, ya que las fotos de caras varían mucho unas a otras (posición, tamaño, ángulo, etc).
- No queríamos escoger un conjunto de datos típico, como puede ser *MNIST* o *IRIS*.
- Entre nuestros ejemplos veremos redes convolucionales. Estas redes funcionan muy bien con problemas de procesamiento de imágenes por lo que este conjunto de datos nos viene genial para los ejemplos.
- No hay demasiados ejemplos (todavía) con TensorFlow. El uso de este conjunto de datos nos permite medir la calidad de nuestro modelo en Kaggle e incentivar el seguir mejorándolo.

Dataset

Más comentarios acerca del dataset:

- Normalizaciones necesarias para las imágenes
- La evaluación del error es por RMSE (por las indicaciones de Kaggle)

Índice

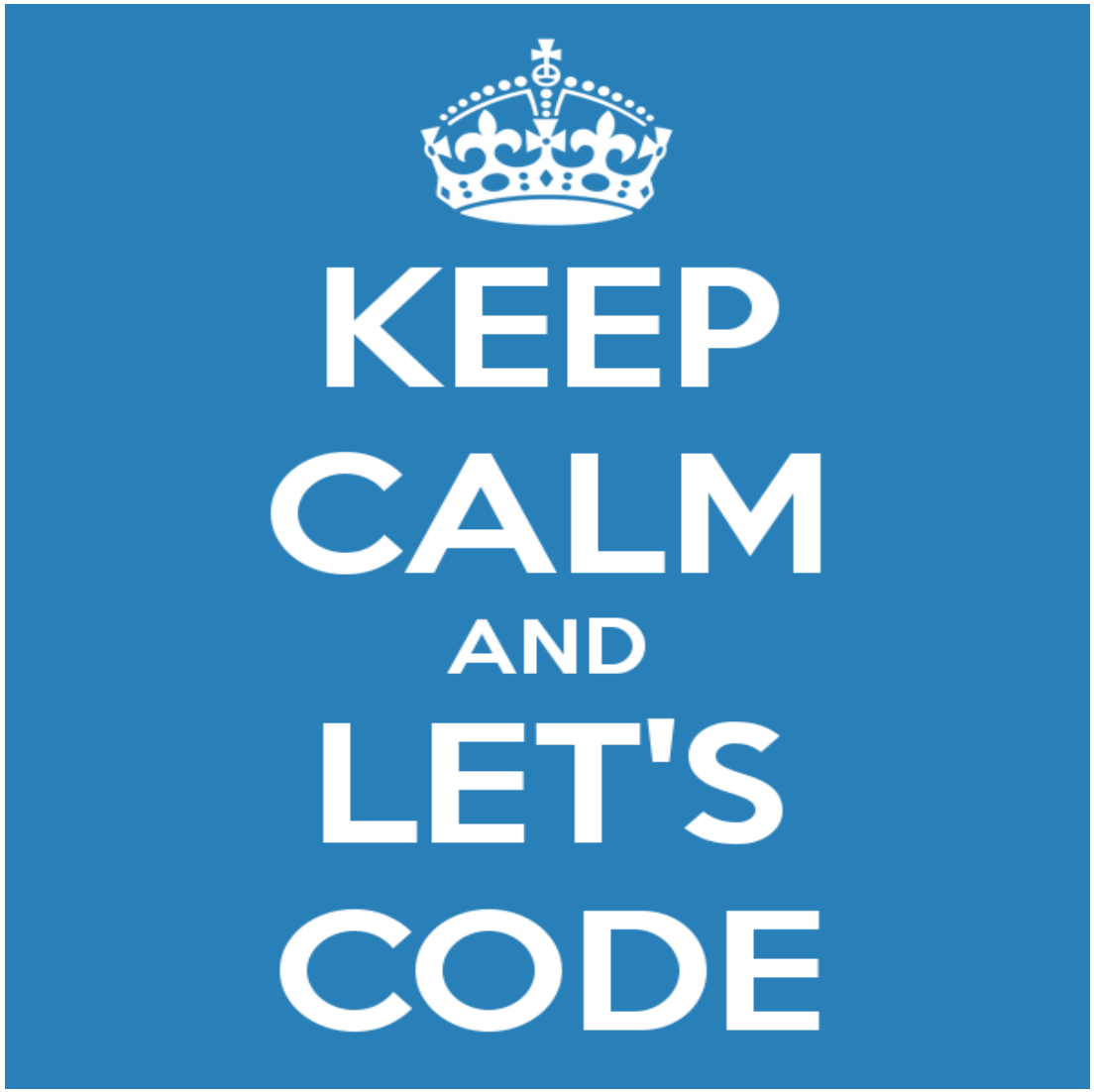
Estructura de los ejemplos

Vamos a ver tres tipos de ejemplos, todos trabajando sobre el conjunto de datos que hemos introducido. Veremos un ejemplo que no usa capa oculta, otro con una capa oculta y un último con redes convolucionales.
La estructura de los ejemplos que veremos la podemos dividir en:

- Importación de librerías
- Tratamiento de datos
- Definición de la estructura del grafo (nodos y tensores)
- Ejecución del grafo (entrenamiento)
- Predicción y resultados

Ejemplos

- Simple NN - Sin capa oculta
- Simple NN - Una capa oculta
- NN convolucional


KEEP
CALM
AND
LET'S
CODE

Comentarios extra

Esto es sólo la punta del iceberg de lo que un framework como TensorFlow nos puede ofrecer.
Por ejemplo, podemos aplicar muchos conceptos y realizar muchas pruebas sobre los parámetros como:

- Probar distintos tipos de optimizadores y de funciones de activación
- Uso de learning rate decay
- Uso de técnicas de regularización, como L2 y dropout
- Infinitud de operaciones que nos permiten mayor o menor control
- Etc, etc, etc

Más comentarios extra

Y además podemos (debemos!) investigar en otras formas de trabajar como puede ser:

- El uso de GPU's y/o operaciones distribuidas
- TensorFlow con Spark (https://databricks.com/blog/2016/01/25/deep-learning-with-spark-and-tensorflow.html)
- TensorFlow como backend de Keras (http://keras.io/backend/)

Más más comentarios extra

- Skflow (https://github.com/tensorflow/skflow)
- TensorBoard
- TensorFlow vs Theano vs Caffe vs ...
- Etc, etc, etc

Índice

Bibliografia

http://www.cs.toronto.edu/ graves/phd.pdf
http://danielnouri.org/notes/2014/12/17/using-convolutional-neural-nets-to-detect-facial-keypoints-tutorial/
https://www.kaggle.com/c/facial-keypoints-detection
https://www.tensorflow.org/
https://github.com/aymericdamien/TensorFlow-Examples