Entrenamiento M1

Para ejecutar el script de entrenamiento se debe llamar de la siguiente forma:
`THEANO_FLAGS=floatX=float32 python run_VAE_hyp.py`
Que ejecuta el archivo indicado pero que en realidad llama al main del archivo `gpulearn_z_x_hyp.py` donde se encuentra todo el código.
Muestra en pantalla cada diez pasos algunas estadísticas como:

- El paso en el que va.
- El tiempo que ha pasado.
- El loglikelihood del batch de entrenamiento.
- El logliklihood del valid set.
- Cuantos pasos lleva sin mejorar.

Respecto a lo último el script se detiene cuando hace 100 pasos sin mejoras. Por experiencia propia hay veces que llega como hatsa 70 pasos sin mejorar y mejora sopresivamente.

Finalmente los resultados se van "mostrando" en la carpeta que se indica al comienzo con "logdir". Básicamente es como qué archivo se está ejecutando, con cuantos nodos ocultos y el tiempo actual. En esa carpeta se generan imágenes que son los samples de la red, a medida que aprende es capaz de generar mejore samples, o sea samples más parecidos con los que entrena.

El código del script está basado en el run_gpulearn_z_x del repositorio original, yo lo limpié, comenté y agregué unos print para saber qué estaba pasando.

Respecto a los datos, escribí una clase para que manejara esto, el archivo se llama hyperspectralData.py. Aquí hay que tener ojo, pues el path que tiene por defecto es el de mi computador, hay que cambiar ese path para que funcione bien. El único problema es que no la he editado para cargar más de un archivo, carga uno solo por defecto.

Dejé a medias (sin hacer en verdad) un método que retornara datos separados por los que están etiquetados y los que no, pero eso es para el modelo de clasificación, no se necesita para la extracción de características.

Ah! Y bueno, algo importante es que este modelo ocupa todos los datos, no solo los etiquetados.

Extracción de características

Para entender como se hace la extracción de características hice un script llamado `extract_features_hyp.py`. Lo iré explicando:

- Primero tenemos que setear el path donde se encuentran los resultados del entrenamiento. En esa carpeta aparte de los samples generados por la red se encuentran los parámetros de la red. Las variables *v* son para la generación de las características, mientras que los parámetros *w* son para generar samples a partir de las variables latentes (características). Hay que tener claro algo aquí:
  - Las características no son algo así concreto o determinista, más bien, siendo rigurosos, son las variables latentes que el modelo aprende, que supuestamente generan los ejemplos reales. La idea es "descubrir" los valores de las variables latentes (que en este caso son distribuciones normales, entonces aprendemos sus parámetros, la media y la varianza) para tener una representación más general de los datos. Es decir, aprendemos como distribuyen estas características más generales, y así, sampleando de estas distribuciones, es capaz de generar nuevos datos, he ahí la razón de generar samples al entrenar el M1.
- Con el path, cargamos las variables del modelo que entrenamos.
- Importamos y cargamos el modelo. Ojo aquí: Hay que setearlos con los mismos hyperparámetros que se ocuparon para entrenar. Los que dejé yo son los que me dieron mejores resultados, pero al ojo, viendo los samples que me generaban, con otros hyperparámetros se generaban imágenes más ruidosas.
- Luego importamos los datos a los cuales les queremos sacar características.
- Después, "transformamos los datos", ocupando la función `dist_qz` que nos entrega la media y la varianza para cada uno de los datos pasados. La función la verdad aún no tengo claro porqué se llama con esos argumentos, pero así la ocupan en el paper y su implementación.
- Finalmente, para extraer las características simplemente sampleamos de las distribuciones, o sea generamos la distribución normal con los parámetros que nos entrega el modelo y sampleamos.