Reporte 10

Jairo Andres Saavedra Alfonso

Universidad de los Andes

1. Objetivos de la semana

- Implementar una red neuronal convolucional para clasificación y regresión.
- Realizar pruebas con capas de Dropout.
- Renormalizar los espectros para entrenamiento.
- Implementar *Learning Rate Scheduler*.
- Implementar *Batch Normalization*.
- Aumentar numero de datos.

2. Clasificación

2.1. Datos

Se utilizaron $80{\cdot}10^3$ datos para todo el proceso de entrenamiento, validación y test. Estos datos están repartidos equitativamente en las cuatro clases principales a clasificar (Estrellas, Galaxias, QSO y QSO BAL). Inicialmente se repartió el conjunto de datos en una relación 80/20 entre *train* y *test*, de los cuales el 25 % de los datos de *train* se asignaron como datos de validación. De esta forma se tienen $48 \cdot 10^3$ espectros para entrenamiento y $32 \cdot 10^3$ espectros entre test y validación.

Todos los espectros usados tienen una dimensión de 886 píxeles en intensidad. Sin embargo, se reduce a la mitad (443 píxeles) y se usa la otra mitad del espectro como peso para la renormalización. El proceso de renormalizaron se lleva a cabo mediante la sustracción del promedio y a este resultado se le divide por el RMS de cada espectro, todo esto usando como peso la variabilidad del espectro restante (de 443 a 886 píxeles).

2.2. Estructura de la red neuronal

En esta semana se continuo con la implementación de un red neuronal convolucional con el objetivo de clasificar espectros. La estructura de la red neuronal convolucional se muestra en la siguiente Tabla 1.

Tabla 1. Arquitectura tentativa de la red neuronal convolucional.

Carpa	Tipo	Característica
Espectro de entrada	Imagen	Espectro con dimensión 1x443 píxeles
conv1	Convolución	Dimensión de entrada 1x443 y salida de 100 con kernel de 10. Stride de 2.
relu	ReLu	Función ReLU
pool	Pooling	Capa de pooling kernel de 2. Stride de 2
conv2	Convolución	Dimensión de entrada 100 y salida de 100 con kernel de 10. Stride de 2.
relu	ReLu	Función ReLU
pool	Pooling	Capa de pooling kernel de 2. Stride de 2
conv3	Convolución	Dimensión de entrada 100 y salida de 100 con kernel de 10. Stride de 2.
relu	ReLu	Función ReLU
pool	Pooling	Capa de pooling kernel de 2. Stride de 2
conv4	Convolución	Dimensión de entrada 100 y salida de 100 con kernel de 10. Stride de 2.
relu	ReLu	Función ReLU
pool	Pooling	Capa de pooling kernel de 2. Stride de 2
dropout	Dropout	Capa de Dropout de 10 %
fc1	Lineal	Lineal: de dimensión de entrada 1800 y dimensión de salida de 16
relu	ReLu	Función ReLU
bn	BatchNorm1d	Capa de Batch Normalization lineal
dropout	Dropout	Capa de Dropout de 10 %
fc2	Lineal	Lineal: de dimensión de entrada 16 y dimensión de salida de 4
log_softmax	Función	Función Sofmax
Capa de salida	Clase	Capa final de valores 0, 1, 2 y 3 respectivos a las clases (Estrella, Galaxia, QSO y QSO BAL)

Se decidió usar como criterio para calcular la inconsistencia entre los valores predichos y los reales la función de perdida (*Loss Function*) ***CrossEntropyLoss***. De igual forma, se uso el algoritmo de optimización de ***Adam*** para actualizar los pesos de la red con base en los datos de entrenamiento.

2.3. Rendimiento de la red neuronal

Con el animo de determinar, independientemente de lo predicho, si la red neuronal esta aprendiendo o no con los datos de entrada. Se evalúa el gradiente de la función de perdida, ***CrossEntropyLoss***, en las épocas que la red fue entrenada. Conforme el numero de entrenamientos aumenta, la función de perdida, una medida cuantitativa de cuanto difiere la predicción del valor real, decrece (Ver Figura 1).

Como se puede ver en la Figura 1, la perdida esta pasando por pequeñas oscilaciones debido a los diferentes *batchs* de entrenamiento. Para cada *batch* nuevo se esta observando que la perdida comienza en un punto mas arriba que el *batch* anterior y desciende conforme se entrena.

Se entreno la red sobre *batchs* de 4000 espectros de test para 10 épocas para los $24 \cdot 10^3$ espectros de entrenamiento. Mediante una matriz de confusión

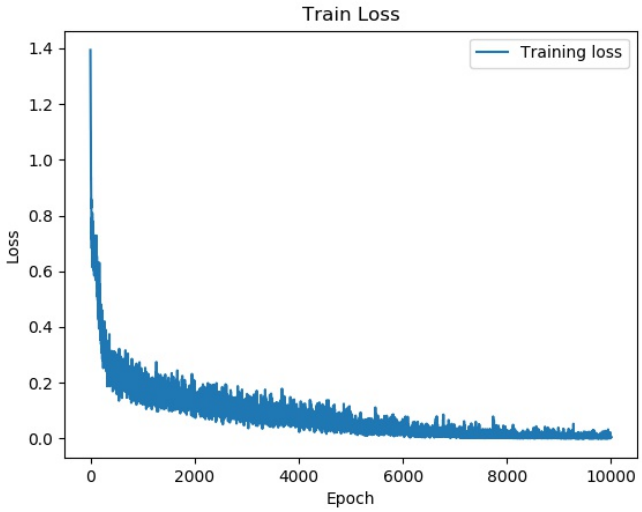

Figura 1. Función de perdida v.s. iteraciones

podemos observar la fracción de espectros clasificados como Estrellas, Galaxias, QSO y QSO BAl con respecto a la etiqueta verdadera, ver Figura 2.

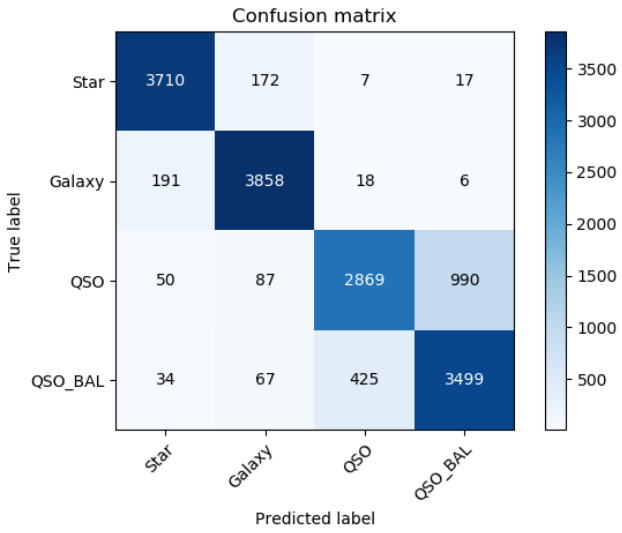

Figura 2. Matriz de confusión para *test*.

Se encontró, después del proceso de entrenamiento, un 84 % de éxito sobre los datos de *test*. De igual forma se evaluó *precision*, *recall* y *F1* del modelo para cada clase. En la Tabla 2 se puede observar los valores de *precision*, *recall* y *F1* para cada clase.

Tabla 2. Valores Precision, Recall y F1 Score para cada clase.

Valor	Estrellas	Galaxias	QSO	QSO BAL
Support	3906	4073	3996	4025
Precision	0.97477	0.94721	0.78441	0.86931
Recall	0.93099	0.92208	0.86441	0.87354
F1	0.95238	0.93447	0.78441	0.81972

Con la implementación de las capas de Dropout, Learning Rate Scheduler y la renormalización de los espectros se logro observar una mejoría en la predicción de clases de 73 % a 93 % de exactitud.

3. Regresión

3.1. Datos

Para la regresión se centro en QSO y QSO BAL siendo los objetos con características mas notables y con amplia distribución de valores de Redshift. Todos los valores de Redshift tienen un etiqueta de confianza del valor reportado (Z_CONF_PERSON) de 1 a 3. Con esta información sobre el Redshift, se seleccionó solo espectros con un valor de confianza de Redshift de 3.

Se realizaron pruebas para espectros de QSO y QSO BAL para los cuales se usaron $80 \cdot 10^3$ y $29 \cdot 10^3$ espectros respectivamente. Al igual que para la tarea de clasificación los espectros están renormalizados.

3.2. Estructura de la red neuronal

En esta semana se continuo con la implementación de un red neuronal convolucional con el objetivo de realizar regresión con los espectros para determinar el Redshift. La estructura de la red neuronal convolucional se muestra en la siguiente Tabla 3 (Semejante a la estructura inicial de clasificación).

Tabla 3. Arquitectura tentativa de la red neuronal convolucional.

Carpa	Tipo	Característica
Espectro de entrada	Imagen	Espectro con dimensión 1x443 píxeles
conv1	Convolución	Dimensión de entrada 1x443 y salida de 64 con kernel de 10. Stride de 2.
relu	ReLu	Función ReLU
pool	Pooling	Capa de pooling kernel de 2. Stride de 1
conv2	Convolución	Dimensión de entrada 64 y salida de 128 con kernel de 10. Stride de 2.
relu	ReLu	Función ReLU
pool	Pooling	Capa de pooling kernel de 2. Stride de 1
conv3	Convolución	Capa de dimensión de entrada 128 y salida de 256 con kernel de 10. Stride de 2.
relu	ReLu	Función ReLU
pool	Pooling	Capa de pooling kernel de 2. Stride de 1
conv4	Convolución	Capa de dimensión de entrada 256 y salida de 256 con kernel de 10. Stride de 2.
relu	ReLu	Función ReLU
pool	Pooling	Capa de pooling kernel de 2. Stride de 1
dropout	Dropout	Capa de Dropout de 50 %
fc1	Lineal	Capa lineal de dimensión de entrada 4608 y dimensión de salida de 128
relu	ReLu	Función ReLU
bn	BatchNorm1d	Capa de Batch Normalization lineal de dimensión 128
fc2	Lineal	Capa lineal de dimensión de entrada 128 y dimensión de salida de 16
relu	ReLu	Función ReLU
dropout	Dropout	Capa de Dropout de 50 %
fc3	Lineal	Capa lineal de dimensión de entrada 16 y dimensión de salida de 1
Capa de salida	Redshift	Valor analítico del Redshift

Se decidió usar como criterio para calcular la inconsistencia entre los valores predichos y los reales la función de perdida (*Loss Function*) ***MSELoss***. De igual forma, se uso el algoritmo de optimización de ***SGD*** para actualizar los pesos de la red con base en los datos de entrenamiento.

3.3. Rendimiento de la red neuronal

QSO

Con el animo de determinar, independientemente de lo predicho, si la red neuronal esta aprendiendo o no con los datos de entrada, se evalúa el gradiente de la función de perdida, ***MSELoss***, en las épocas que la red fue entrenada. Conforme el numero de iteraciones aumenta se puede observar como decrece la función de perdida (Ver Figura 3). Esto nos indica que en cada iteración la red esta detectado características de los espectros de QSO de forma que se puede decir que la red esta "aprendiendo".

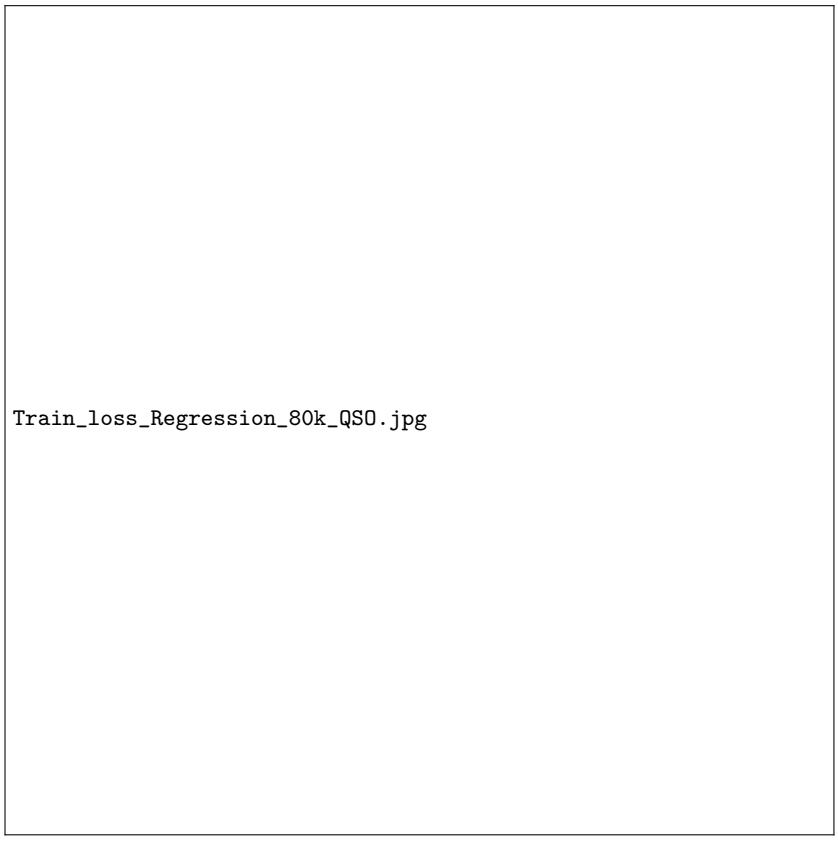

Figura 3. Función de perdida v.s. iteraciones para regresión de Redshift de QSO

Para evaluar el rendimiento de la red se calculo el valor de **MSE** y **MAE** para un conjunto de test de 16^3 espectros. Se encontró un valor de MSE y MAE de *tanto* y *tanto* respectivamente.

QSO BAL

De igual forma que para QSO BAL se evaluó el rendimiento de la red 3. El proceso de test se llevo a cabo para 16^3 espectros. De igual forma se observa como decrece la función de perdida con respecto a las iteraciones (Ver Figura 4).

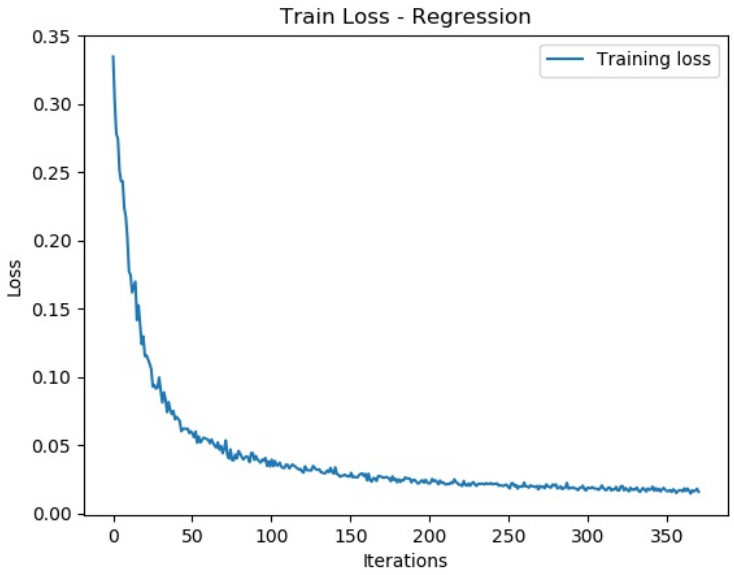

Figura 4. Función de perdida v.s. iteraciones para regresión de Redshift de QSO BAL

Para evaluar el rendimiento de la red se calculo el valor de **MSE** y **MAE** para un conjunto de test de 16^3 espectros. Se encontró un valor de MSE y MAE de 1.72 % y 10.1 % respectivamente.

4. Tareas pendientes

- Profundizar sobre el funcionamiento, estructura y evaluación de redes neuronales convolucionales.
- Mejorar implementación para regresión.
- Aumentar el número de datos para entrenamiento mediante el uso del HP cluster.
- Paralelizar todo.

5. Repositorio Github

En este repositorio[1] se pueden ver todos los reportes semanales y el notebook.

Referencias

1. Busca G., Balland C., (2018).: QuasarNET: Human-level spectral classification and redshifting with Deep Neuronal Networks. Retrieve from: https://arxiv.org/abs/1808.09955
2. Paris, I., et al., (2017), Astron. Astrophys..: The Sloan Digital Sky Survey Quasar Catalog: Twelfth data release. , 597, A79

[1] https://github.com/MrX1997/Reportes-Proyecto-de-Monografa/blob/master/Noteboooks/SpectraNET.ipynb