Proyecto de Red Neuronal Recurrente

RNN

Curso: Redes Neuronales Profundas
Maestría en Ciencia de Datos
Universidad de Sonora

Contenido:

Repositorio: https://github.com/BrendaLPhys/LSTM-RNN

1. Selección de datos secuenciales

Erupciones Solares

Las condiciones dinámicas y eventos en el sol en el espacio cercano a la Tierra y en la parte alta de la atmósfera juegan un papel importante tanto en las vidas humanas como en la tecnología. Estos fenómenos son descritos por el clima espacial. La actividad solar da lugar a variedad de climas espaciales. La actividad solar consiste en erupciones solares, eyecciones de masa coronal o CME (por sus siglas en inglés: Coronal Mass Ejection), viento solar de alta velocidad, y partículas energéticas solares. El campo solar magnético es la principal fuente de esta actividad. **[1]**

Las erupciones solares son erupciones energéticas de radiación electromagnética del sol con una duración de entre sólo unos minutos hasta horas. Los impactos terrestres de erupciones pequeñas son limitados, pero erupciones fuertes tienen efectos significativos en la atmósfera. El aumento en la ionización afecto la cantidad de electrones, que a su vez afecta la transmisión de ondas de radio y la exactitud de sistemas de posicionamiento globales. El calentamiento ionosférico hace que se expanda la atmósfera, aumentando la densidad de la tierra y el arrastre de satélites, alterando sus órbitas. Erupciones fuertes están comunmente acompañadas de eyecciones de masa coronal o CMEs que causan un impacto considerable en el medio ambiente de la Tierra. Por lo tanto, vale la pena mejorar la predicción de erupciones solares, especialmente las de mayor intensidad. **[2]**

Los datos utilizados provienen del producto llamado Spaceweather HMI Active Region Patches, producido por el equipo SDO/HMI. Estos datos fueron liberados a finales de 2012 y pueden ser encontrados como la serie de datos hmi.sharp en el *Joint Science Operations Center (JSOC)*. Los datos de SHARP comprenden regiones activas automáticamente identificadas y rastreadas en mapas y proporcionan una buena cantidad de parámetros físicos útiles para la detección de erupciones. **[3]**

TOTUSJH = Total unsigned current helicity
Total unsigned magnetic helicity para las regions activas muestra una tendencia en aumento antes de la presencia de una primera erupción y la región activa comienza a tener actividad de erupciones cuando este valor es lo suficientemente alto.

Se decide trabajar con la predicción del comportamiento de esta variable.

Referencias y fuentes de consulta:
•Artículo 1 - Forecasting Space Weather Using Deep Learning Techniques **(2018)**
•Artículo 2 - Predicting Solar Flares with Machine Learning: Investigating Solar Cycle Dependence **(2020)**
•Artículo 3 - Predicting Solar Flares Using a Long Short-term Memory Network **(2019)**

2. Procesamiento de datos

El dataset original contiene información sobre erupciones solares, la hora o *timestamp* y una serie de variables relacionadas a las características magnéticas de la superficie solar.

	label	flare	timestamp	NOAA	HARP	TOTUSJH	Cdec	TOTUSJZ	Chis1d	USFLUX	...	Bhis1d	Bhis	Mhis1d	EPSZ	MEANGBH	MEANJZH
0	Negative	N	2010-05-03T05:34:22.60Z	11063	11	126.760	0.0	56.667	0	41.194717	...	0	0	0	-0.0554	64.861	43.057
1	Negative	N	2010-05-03T06:34:22.60Z	11063	11	123.469	0.0	44.136	0	39.738983	...	0	0	0	-0.1693	71.024	37.093
2	Negative	N	2010-05-03T07:34:22.60Z	11063	11	135.282	0.0	50.456	0	40.148766	...	0	0	0	-0.1909	73.637	32.714
3	Negative	N	2010-05-03T08:34:22.60Z	11063	11	110.462	0.0	33.387	0	40.451828	...	0	0	0	-0.1900	74.494	29.658
4	Negative	N	2010-05-03T09:34:22.60Z	11063	11	120.256	0.0	30.005	0	42.458454	...	0	0	0	-0.1687	76.406	28.702

De este se seleccionan la variable temporal y los valores para Total unsigned current helicity .

	timestamp	TOTUSJH
0	2010-05-03 05:34:22.600000+00:00	126.760
1	2010-05-03 06:34:22.600000+00:00	123.469
2	2010-05-03 07:34:22.600000+00:00	135.282
3	2010-05-03 08:34:22.600000+00:00	110.462
4	2010-05-03 09:34:22.600000+00:00	120.256

Tomando solo el día:

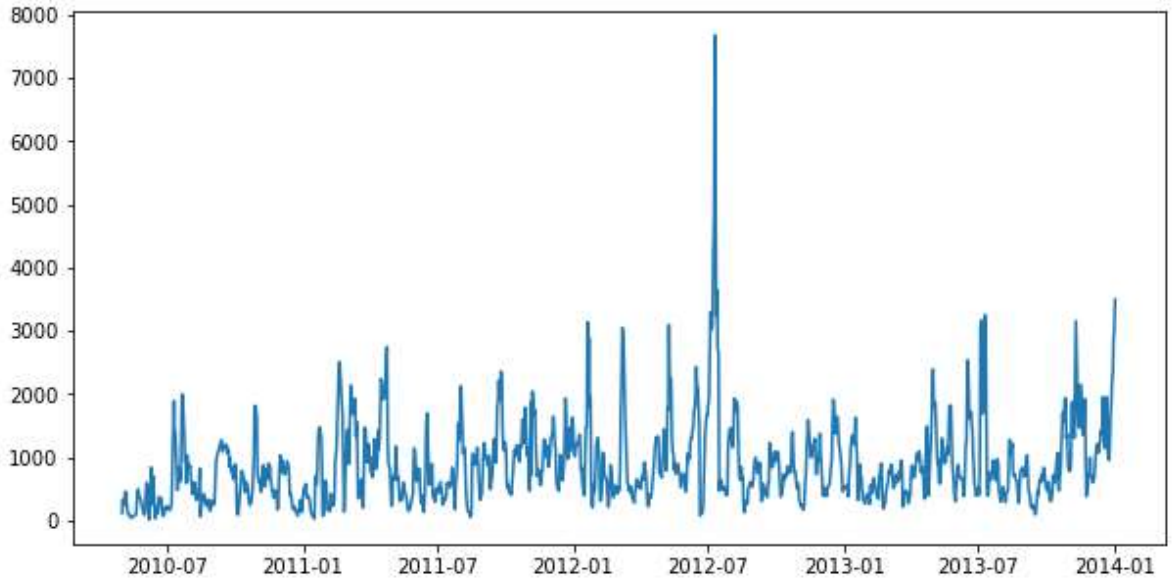
	timestamp	TOTUSJH
0	2010-05-03	126.760
1	2010-05-03	123.469
2	2010-05-03	135.282
3	2010-05-03	110.462
4	2010-05-03	120.256
...	...	...

Y generando el valor promedio por día de la variable:

	TOTUSJH
timestamp	
2010-05-01	115.969200
2010-05-02	321.134385
2010-05-03	235.476355
2010-05-04	253.746148
2010-05-05	443.006350

Observamos el comportamiento del valor diario promedio en el tiempo:

```
plt.rcParams["figure.figsize"] = (10,5)
plt.plot(df.TOTUSJH);
```



Normalizamos los datos, separamos entrenamiento y prueba en 67 y 33 por ciento respectivamente, y se generan los arreglos que definen la secuencia temporal y establecen la interacción entre el tiempo t y el t+1.

```
# normalize the dataset
scaler = MinMaxScaler(feature_range=(0, 1))
df = scaler.fit_transform(df)
```

```
# split into train and test sets
train_size = int(len(df) * 0.67)
test_size = len(df) - train_size
train, test = df[0:train_size,:], df[train_size:len(df),:]
print(len(train), len(test))
```

```
879 434
```

```
# convert an array of values into a dataset matrix
def create_dataset(dataset, look_back=1):
    dataX, dataY = [], []
    for i in range(len(dataset)-look_back-1):
        a = dataset[i:(i+look_back), 0]
        dataX.append(a)
        dataY.append(dataset[i + look_back, 0])
    return np.array(dataX), np.array(dataY)
```

```
# reshape into X=t and Y=t+1
look_back = 1
trainX, trainY = create_dataset(train, look_back)
testX, testY = create_dataset(test, look_back)
```

```
# reshape input to be [samples, time steps, features]
trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
```

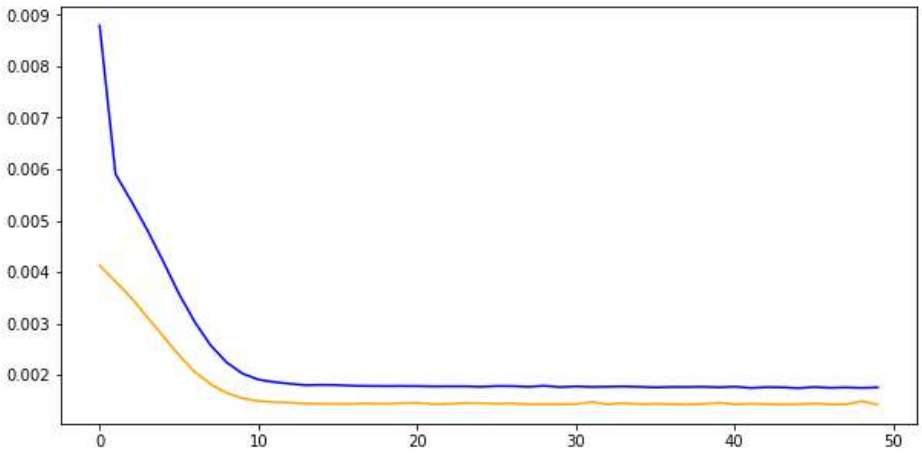
3. Red neuronal

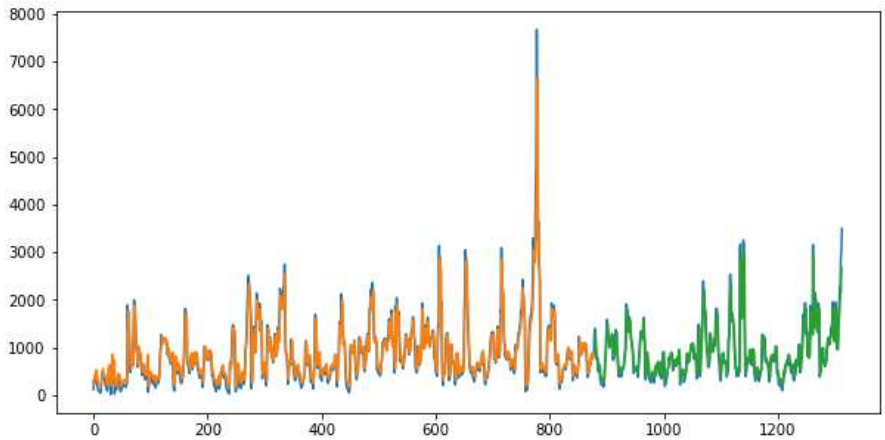
```
# create and fit the LSTM network
model = Sequential()
model.add(LSTM(4, input_shape=(1, look_back), activation='relu'))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
history = model.fit(trainX, trainY, validation_data=(testX, testY), epochs=50, batch_size=5, verbose=2)
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(loss))
```

La red neuronal sigue un modelo secuencial de Keras sencillo con una capa LSTM, adam como optimizador y una pérdida definida por el error cuadrático medio. El *batch_size* seleccionado es de 5.

4. Detección de posible *overfitting*

Las curvas obtenidas con el modelo de Red Neuronal *baseline* no muestran evidencia de sobreaprendizaje.





5. Variaciones de la Red Neuronal

5.1 Red Bidireccional y *Stacked*

Se emplea una red Bidireccional conservando los hiperparámetros de la red *baseline*.

```
# create and fit the LSTM network
model = Sequential()
model.add(Bidirectional(LSTM(4, input_shape=(1, look_back), activation='relu')))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
history = model.fit(trainX, trainY, validation_data=(testX, testY), epochs=50, batch_size=5, verbose=2)
loss = history.history['loss']
epochs = range(len(loss))
```

Se emplea una red *Stacked* conservando los hiperparámetros de la red *baseline*.

```
# create and fit the LSTM network
model = Sequential()
model.add(LSTM(4, input_shape=(1, look_back), activation='relu', return_sequences=True,))
model.add(LSTM(4, activation='relu'))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
history = model.fit(trainX, trainY, validation_data=(testX, testY), epochs=50, batch_size=5, verbose=2)
loss = history.history['loss']
epochs = range(len(loss))
```

5.2 Cambio de Optimizador

Seleccionamos el optimizador *RMSProp* (Root Mean Square Propagation) conservando los hiperparámetros de la red baseline.

```
# create and fit the LSTM network
model = Sequential()
model.add(LSTM(4, input_shape=(1, look_back), activation='relu'))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='RMSProp')
history = model.fit(trainX, trainY, validation_data=(testX, testY), epochs=50, batch_size=5, verbose=2)
loss = history.history['loss']
epochs = range(len(loss))
```

5.3 Cambio de *batch_size*

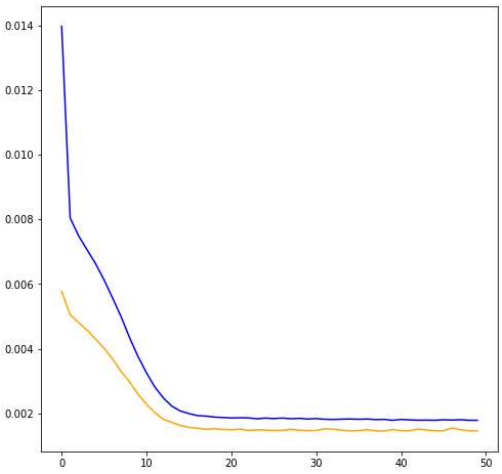
Se emplea un *batch_size de 10* que es el doble de tamaño del previamente usado y conservando los hiperparámetros de la red *baseline*.

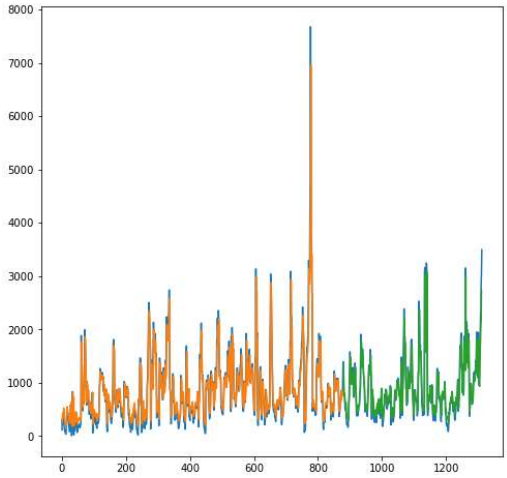
```
# create and fit the LSTM network
model = Sequential()
model.add(LSTM(4, input_shape=(1, look_back), activation='relu'))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
history = model.fit(trainX, trainY, validation_data=(testX, testY), epochs=50, batch_size=10, verbose=2)
loss = history.history['loss']
epochs = range(len(loss))
```

6. Resultados

Baseline

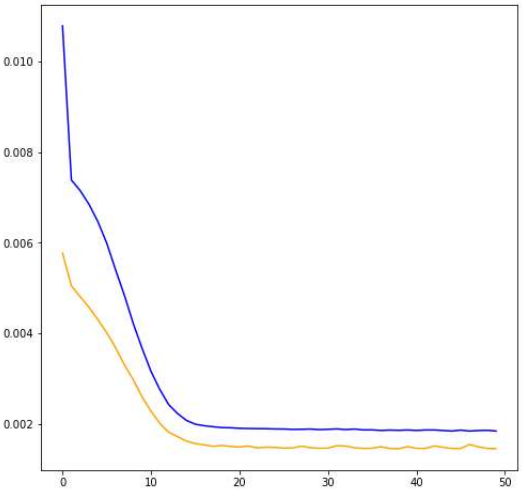
Train Score:
0.04 RMSE
Test Score:
0.04 RMSE

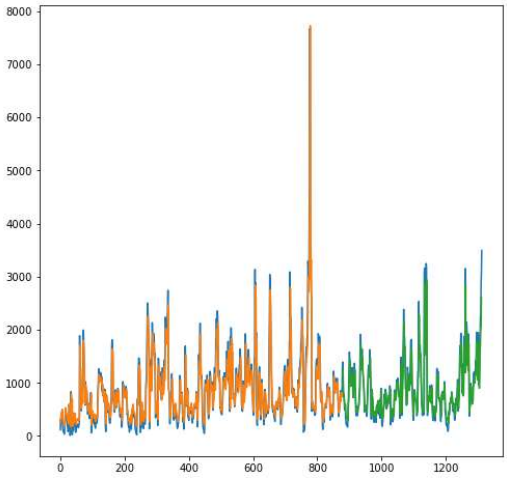




Bidireccional

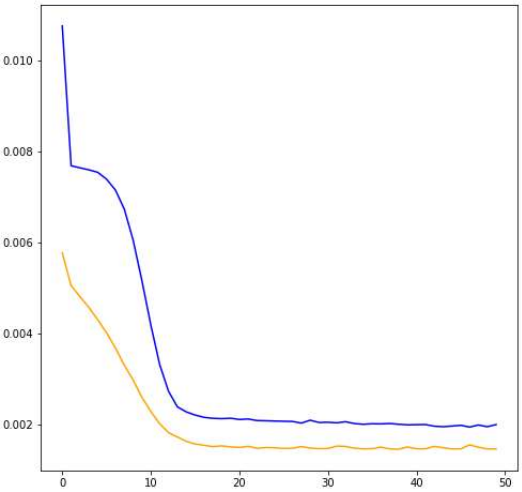
Train Score:
0.04 RMSE
Test Score:
0.04 RMSE

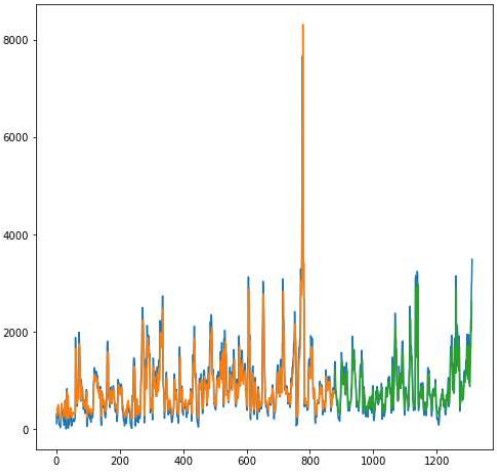




Stacked

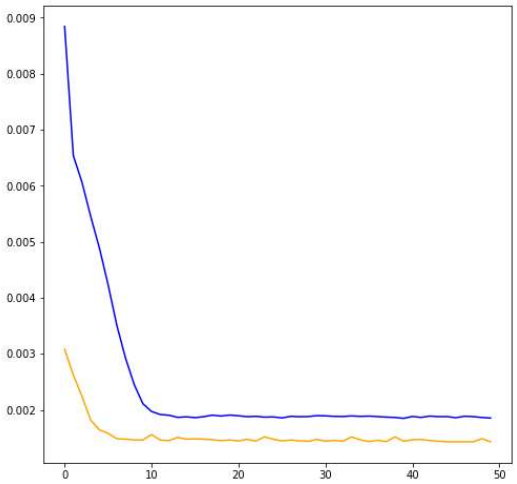
Train Score:
0.05 RMSE
Test Score:
0.04 RMSE

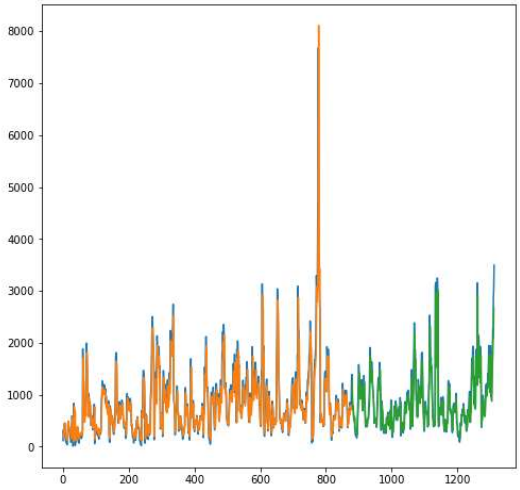




RMSprop

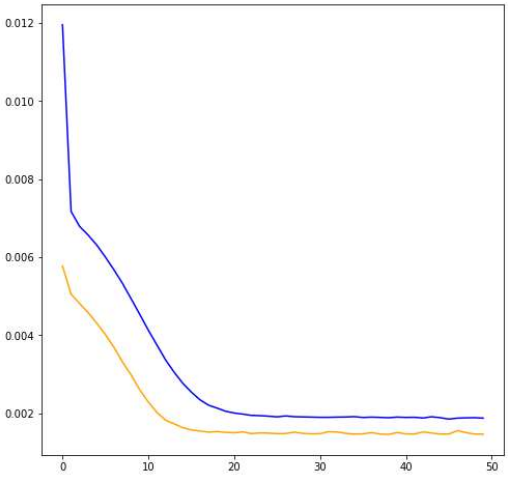
Train Score:
0.04 RMSE
Test Score:
0.04 RMSE

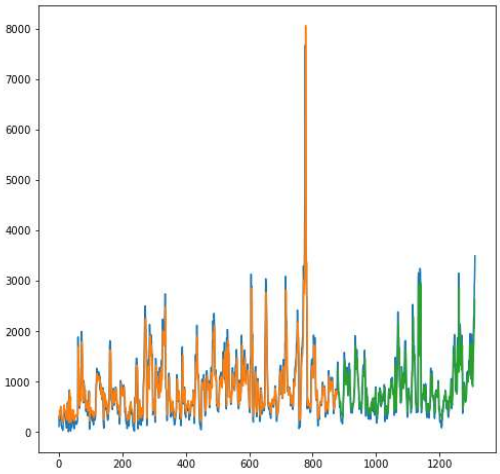




batch_size of 10

Train Score:
0.04 RMSE
Test Score:
0.04 RMSE





7. Conclusiones

El error RMSE Root Mean Squared Error, se comportó de manera equivalente en todos los modelos explorados. Las curvas de comportamiento de pérdida en entrenamiento y prueba también son muy similares para las variaciones con las cuales se probó el algoritmo.

El siguiente paso sería hacer combinaciones de estos cambios en busca de un comportamiento óptimo, sin embargo se considera que el entrenamiento fue efectivo, esto se puede apreciar de manera gráfica en las curvas de tipo serie de tiempo en donde se encuentran los datos originales y las predicciones del modelo en entrenamiento y prueba.

El modelo logra ajustarse al comportamiento de los datos originales por lo que se puede ser optimista en el uso de este tipo de algoritmos para describir el comportamiento de la variable magnética seleccionada, ayudando así a tener la posibilidad de predecir su comportamiento futuro.

Cabe recalcar que la descripción de tipo serie de tiempo se hizo de los valores promedio diarios de la variable, sin considerar las fluctuaciones que esta puede tener a una menor escala. Sin embargo, este ejercicio se llevó a cabo con fines demostrativos y como una manera de poder conocer mejor el uso de las redes neuronales recurrentes de tipo LSTM.