Deep Learning: Recurrent Neural Network № 7

Daniela Pinto Veizaga, dpintove@itam.mx
Diego Villa Lizárraga, dvillali@itam.mx

12/03/2020

Introducción

El propósito de esta tarea es aprender sobre las redes neuronales recurrentes, en general; y, en específico, sobre la red" larga memoria a corto plazo" (LSTMs, por sus siglas en inglés): un subtipo de red neuronal recurrente capaz de aprender de las dependencias de largo plazo.

Para el desarrollo de esta tarea, nos basamos en el ejercicio original realizado por Asutosh Nayak, disponible en el siguiente sitio, dónde el autor predice los precios de *stock*, dado un dataset.

Contamos con un dataset con 14,058 observaciones y 7 variables o columnas. Para la implementación del primer modelo, dividimos la base de datos de la siguiente manera: *training data* (11,246) y *test data* (2,812).

Pregunta 1

Renombra los ndarrays A, B, C, usando la notación vista en clase: $W(h)$, $W(x)$, etc...

Conforme con las instrucciones, procedemos con el cambio de las etiquetas A, B, C por la notación vista en clases: $W(h)$, $W(x)$ y $W(y)$.

Listing 1: Renaming the labels

```
W_xh, W_hh, W_yh = rnn_model.layers[0].get_weights()
print(W_xh.shape)
print(W_hh.shape)
print(W_yh.shape)
```

Para este cambio de etiquetas nos basamos en el a el siguiente link: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Pregunta 2

El rendimiento es excelente, sin embargo, hay un problema de dependencia en los datos. ¿Puedes notar cuál es, y explicar el excelente rendimiento?

En general, el excelente rendimiento se debe a que estamos incluyendo la variable que buscamos predecir ('close' – X train[:, 3]) en las variables explicativas. Esto se conoce como d*data leaking*, es decir, estamos utilizando información que en un escenario real no tendríamos.

Más concretamente, al formatear los datos –para convertir la serie de tiempo a un set de pares (x, y) que pueda alimentar la Red Neuronal Recurrente (RNN)–, se genera simultáneamente $y[ind]$ y $x[ind]$. Lo peculiar es que $x[ind]$, incluye un elemento que forma parte de $y[ind]$.

Listing 2: Dependencia de datos

```
1
2   for ind in range(n_samples):
3     x[ind] = X[ind : ind + hist_size]
4     y[ind] = Y[ind + hist_size]
5   return x, y
```

Con el proceder anterior, basta para tener buenos rendimientos puesto que ocurre una especie de cheating en la predicción de $y[ind]$: se utilizan varios features, entre los cuáles uno es igual a $y[ind]$.

Pregunta 3

Crea una función "delayed_dataset" que formatee los datos para asociar $y[t]$ *con* $x[t - delay]$ *hasta* $x[t-delay-history]$*. Por ejemplo, para hacer prónosticos del tipo* $y[10] = f(x[7], x[6], x[5]; \Omega)$*. Donde tanto* x *como* y *pueden ser univariados o multivariados.*

Listing 3: Función delayed_dataset

```
1
2  # AUX funtion to build time-series dataset
3  def delayed_dataset(X, Y, delay, hist_size):
4    '''
5    Params
6      X: data matrix [n_time_steps, n_X_feats]
7      Y: label matrix [n_time_steps, n_Y_feats]
8      delay: integer indicating the delay between last time step in x ↩
           and time step in y
9      hist_size: integer indicating the number of time steps in each ↩
           sample of x
10   Returns
11     x: tensor of input data [n_samples, n_timesteps, n_X_features]
12     y: tensor of output data [n_samples, n_Y_features]
13   '''
14   n_samples = Y.shape[0] - hist_size - delay
15   x = np.zeros((n_samples, hist_size, X.shape[1]))
```

```
  y = np.zeros((n_samples,Y.shape[1]))

  for ind in range(n_samples):
    x[ind] = X[ind: ind + hist_size]
    y[ind] = Y[ind + hist_size + delay]
  return x, y
```

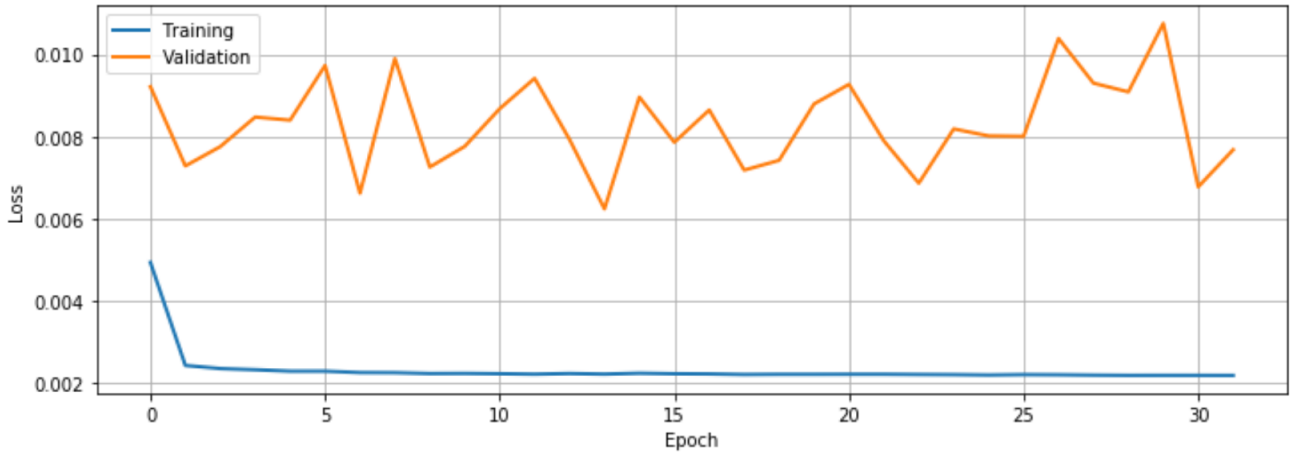
Pregunta 4

Diseña una RNN para resolver el problema de regresion multivariada que acabas de generar. Reporta el mejor resultado que hayas obtenido.

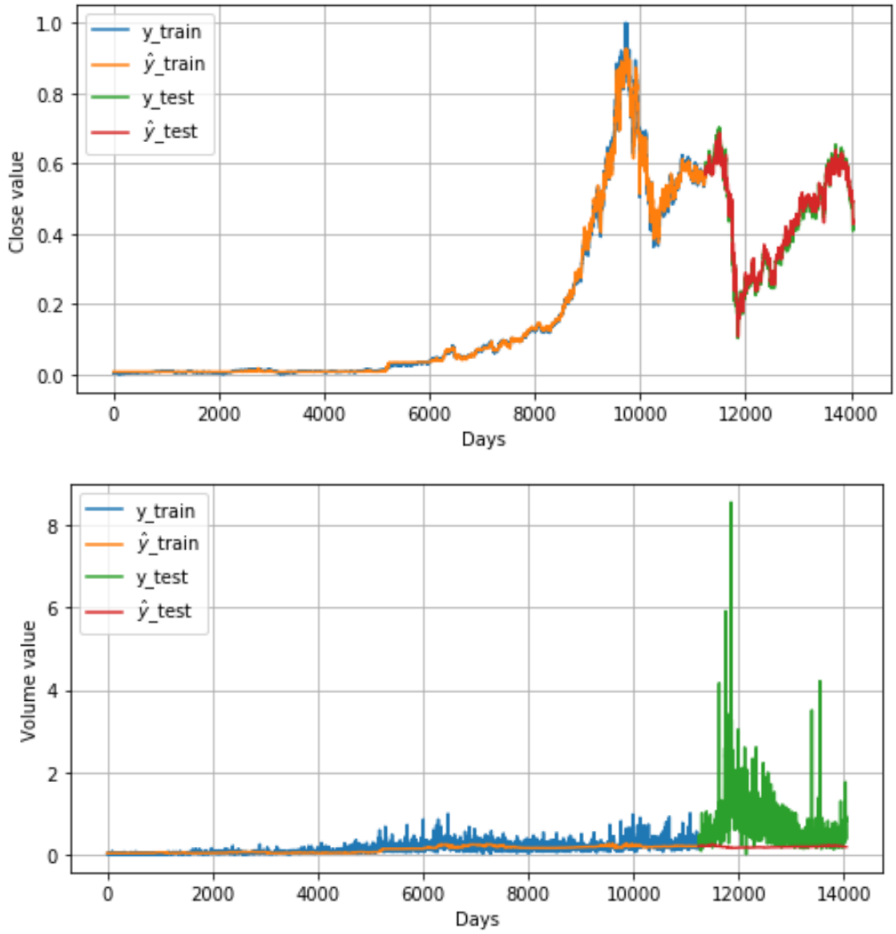
Se implementó un red recurrente de 2 capas, la primera capa es una SimpleRNN de 32 nodos y la segunda es una capa densa de dos nodos con activación sigmoide. Utilizamos una función de perdida "Mean squared error" y un optimizador "rmsprop".

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_1 (SimpleRNN)     (None, 32)                1152
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 66
=================================================================
Total params: 1,218
Trainable params: 1,218
Non-trainable params: 0
_________________________________________________________________
```

Gráfica 1. Diseño RRN



Gráfica 2. Grafica Epocas vs Loss

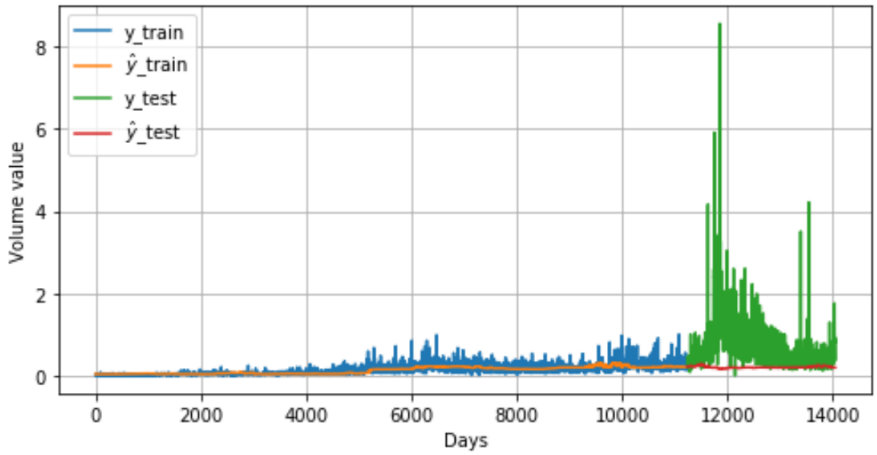


Gráfica 3. Evaluación

Pregunta 5

Parece que predecir 'close' es muy fácil, pero no predecir 'volume'. Explica las razones por las cuales crees que obtienes esos resultados.

En general, se puede observar en la gráfica 3. de "Volumen" que el rango de valores del volumen de y_test (después de 11,000 días) son atípicos comparado contra el rango de los valores de y_train que son con los que se entrenó la red. Gracias a ésto el $\hat{y}\_test$ mantiene el mismo piso que los valores con los que fue entrenado trayendo como consecuencia una brecha importante entre $\hat{y}\_test$ y y_test.



Gráfica 4. Volumen

En específico, predecir *close* es más fácil que predecir volume por las siguientes razones:

1. Las variables correspondientes a los **precios**: open, high, low y close, con tendencias similares que permiten que la red recurrente mejore su pronóstico. En cambio, la serie correspondiente a **volumen** no incorpora tendencia, lo que hace que el único feature que incorpore dicha conducta sea el valor rezagado de volumen. Dado lo anterior, creemos que esto afecta de forma negativa la predicción hecha por la red neuronal en cuanto a dicha variable.
2. Las escalas de los *features* incorporados en los **precios** y **volumen** son distintos. Dado lo anterior, el mejor predictor del volumen es esta misma variable rezagada, mientras que el resto de features incorporados en la predicción de **volumen** perjudican la predicción.

Pregunta 6

Ahora encuentra la forma de pasar columnas no consecutivas a la función "delayed_dataset". Crea sets donde, x contenga (open, low, volume), e y contenga (high, close). Considera delay=3 y history_size=4.

La forma de pasar columnas no consecutivas a la función $delayed\_dataset$ es mediante una tupla con los índices de las columnas deseadas.

En este caso, para introducir dentro de los features las variables: $open(1)$, $low(1)$ y $volumen(1)$, se utiliza la tupla: $(1, 1, 1)$. Para definir dentro de las variables objetivo: $high(1)$ y $close(3)$, se utiliza la tupla: $(1, 3)$.

Considerando $delay = 3$ y $history\_size = 4$, tenemos el siguiente código:

Listing 4: Tuplas

```
x_train, y_train = delayed_dataset(X_train[:,(1,1,1)], X_train[:, ↩
    (1,3)], delay=3, hist_size=4)
x_test, y_test   = delayed_dataset( X_test[:,(1,1,1)], X_test[:, ↩
    (1,3)], delay=3, hist_size=4)
```

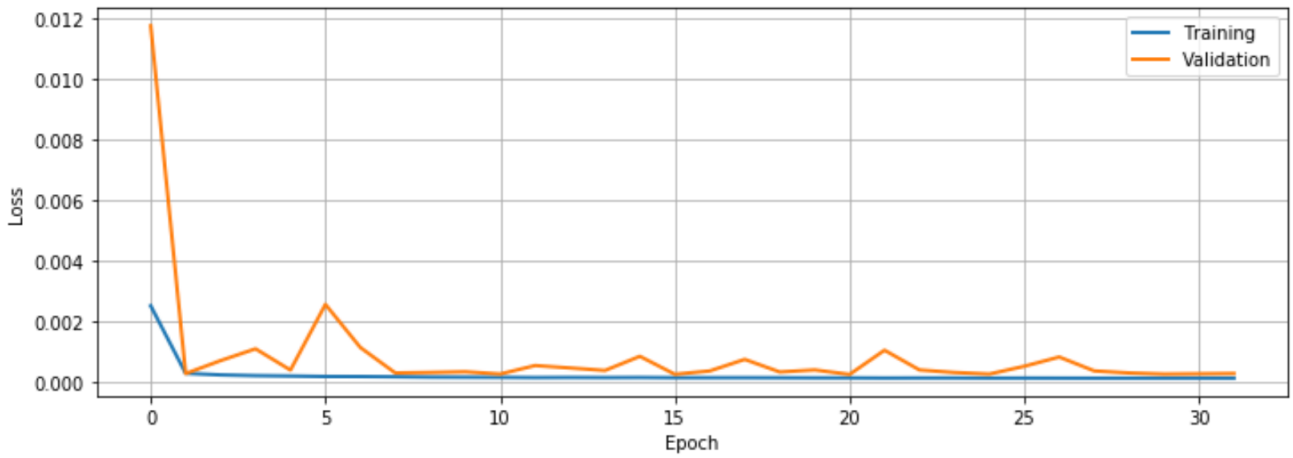
Pregunta 7

Diseña una red para resolver este último problema de regresión. Reporta tu red con el mejor resultado.

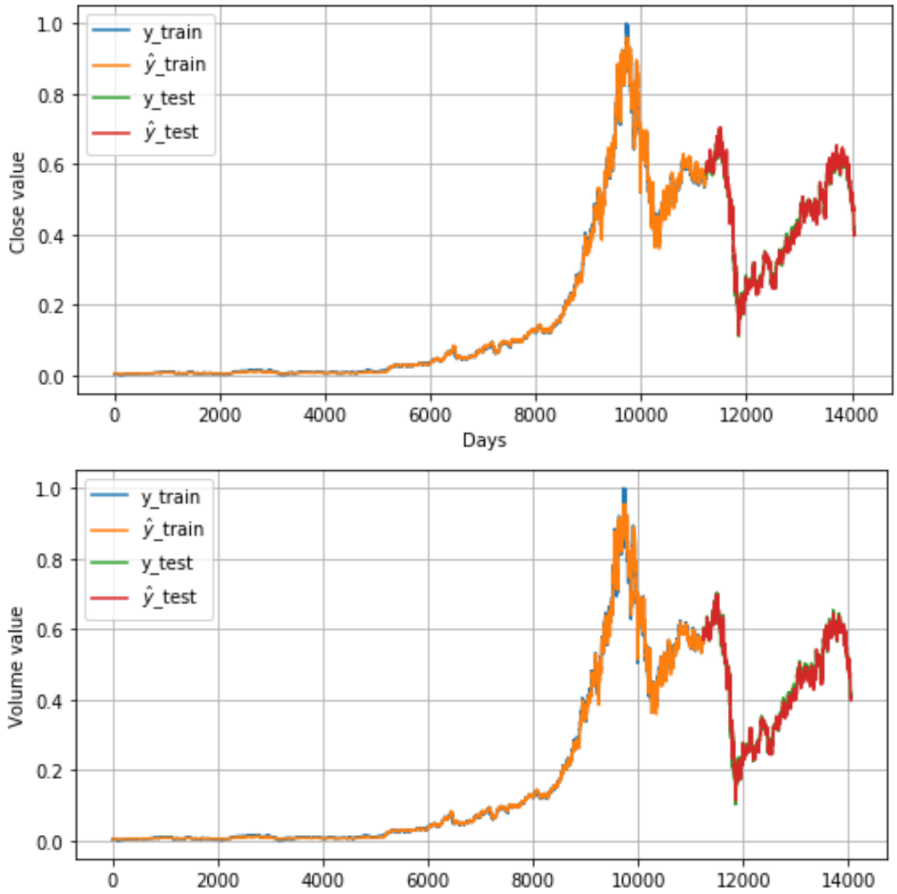
Se implementó un red recurrente de 3 capas, la primera capa es una SimpleRNN de 32 nodos, la segunda es una capa densa de 32 nodos con activación ReLU y la tercera es un capa densa de dos nodos con activación sigmoide. Utilizamos una función de perdida "Mean squared error" y un optimizador "rmsprop".

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_2 (SimpleRNN)     (None, 32)                1152
_________________________________________________________________
dense_2 (Dense)              (None, 32)                1056
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 66
=================================================================
Total params: 2,274
Trainable params: 2,274
Non-trainable params: 0
_________________________________________________________________
```

Gráfica 5. Diseño RRN



Gráfica 6. Grafica Epocas vs Loss



Gráfica 7. Evaluación