Informe Proyecto MA5204: Aprendizaje de Máquinas
Modelos de series de tiempo para predecir contaminantes en el aire.

Pablo Araya [1] **Jose Cornejo** [1] **Bruno Hernández** [1] **Javier Santibáñez** [1]
Felipe Tobar [2]

1. Introducción

Se estudia el comportamiento de contaminantes en el aire con series de tiempo. Esto con el objetivo de poder predecir las concentraciones de estos contaminantes a futuro. Para esto se estudian dos modelos: ARIMA y redes neuronales. Se analiza la predicción inmediata y a más pasos en el futuro, además de una comparación entre ambos modelos. (Video)

2. Descripción de los datos

La base de datos con la cual se trabaja fue obtenida del repositorio dedicado al aprendizaje de máquinas de la Universidad Pública de Irvine, California. Esta base fue construida a partir de un dispositivo multisensor de gas en una ciudad italiana. Este sensor guarda los promedios de concentraciones cada hora de 5 químicos de óxido de metal, de concentraciones en la tierra y también de datos meteorológicos. Los datos fueron obtenidos desde marzo de 2004 hasta febrero de 2005, dando un total de 9358 observaciones y 15 atributos.

De las variables de interés se tienen los químicos CO (monóxido de carbono), NO2 (dióxido de nitrógeno), NOx (total de óxidos de nitrógenos) y NMHC (hidrocarburos no metánicos). Además, se tienen estos datos para dos valores: el primero representado con GT que corresponde a datos confiables proporcionados por un sensor de una agencia italiana de protección ambiental, el segundo representado por PT corresponde a los datos obtenidos por el multisensor registrado a través de la resistencia eléctrica. En lo que sigue se trabaja con los datos asociados a PT.

3. Procesamiento de los datos

Al cargar los datos, lo primero que se realiza es cambiar el formato de los datos a formatos adecuados, es decir, que las fechas estén guardadas en formato fecha y los valores de las concentraciones de forma numérica. Luego, se estudian los valores faltantes. En las variables de interés se tiene que hay 366 datos faltantes y en las otras (GT) se tienen entre 360 y 1600 datos faltantes.

Se decide usar la relación entre las columnas GT y PT para rellenar los datos faltantes. Para esto se grafican las columnas y se observa que para las variables CO y NOx existe una relación lineal y exponencial respectivamente entre los datos. Por lo tanto se realiza regresión lineal para la variable CO y para NOx se ajusta una curva exponencial, como se puede observar en la Figura 3.1. Con estos modelos se procede a llenar los datos faltantes para estas variables, para el resto de las variables de interés se llenan mediante el método *ffill* de pandas que hace un llenado *hacia adelante* (*forward*) de los datos faltantes lo cual es bastante útil dado que se está trabajando con series de tiempo.

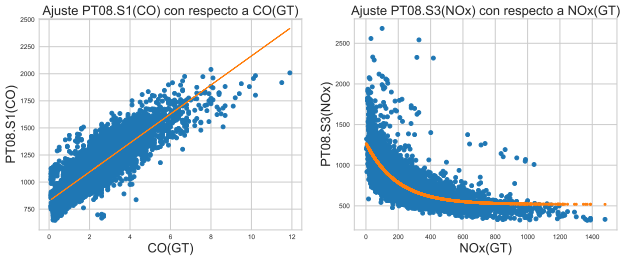

Figura 3.1. Ajustes de CO y NOx

4. ARIMA

4.1. Marco Teórico

El modelo *Autoregressive Integrated Moving Average* (ARIMA) es un modelo de regresión lineal que es utilizado bastante para la predicción de series de tiempo. Este utiliza variaciones y regresiones de datos estadísticos con el fin de encontrar patrones para una predicción hacia el futuro. El modelo tiene la siguiente estructura:

$$\left(1-\sum_{i=1}^{p}\phi_i L^i\right)(1-L)^d X_t = \left(1-\sum_{i=1}^{q}\theta_i L^i\right)\varepsilon_t.$$

Donde L es el operador *shift* ($LX_t = X_{t-1}$) y los parámetros ϕ_i, θ_i son determinados por este modelo de tal manera que minimicen el error cuadrático medio. Los parámetros $p, q, d \in \mathbb{N}$ se determinan según una lista de posibles valores y se elige el que minimice el Criterio de Información de Akaike. El parámetro p representa el número de datos anteriores que se usa para predecir, el parámetro q corresponde al número de errores anteriores que se usan y el

[1]Integrantes [2]Profesor .

parámetro d hace referencia al mínimo número de diferenciación necesario para hacer la serie estacionaria. La forma de referirse a estos modelos es de la forma ARIMA(p, d, q).

4.2. Resultados y Análisis

Como primer acercamiento se decide estudiar el desempeño del modelo para la variable CO, ya que para las otras es similar. En este caso se utiliza una extensión del modelo llamado SARIMA que considera una componente de *seasonality* o estacionalidad en los datos. Para los parámetros del modelo se usa $p = 24$ ya que con esto se toman los datos de las 24 horas anteriores para hacer la predicción, $d = 0$ ya que los datos parecen ser estacionarios y para determinar el valor de q se hace una búsqueda con `auto_arima` de la librería `pmdarima.arima` para valores entre 1 y 7, obteniendo finalmente $q = 6$.

Además, para estudiar el desempeño del modelo en los datos, estos se separan en dos conjuntos: uno de entrenamiento y uno de prueba (test). Al tratarse de series temporales esta separación no puede ser aleatoria por lo que se separan hasta cierto momento. Así, el conjunto de entrenamiento corresponde a los primeros 7485 datos y el de test a los 1872 siguientes.

El rendimiento del modelo en el conjunto de entrenamiento se muestra en la Figura 4.1 donde solo se muestran los primeros 300 datos. La predicción se ajusta de buena manera a los datos y además sigue la tendencia de los datos.

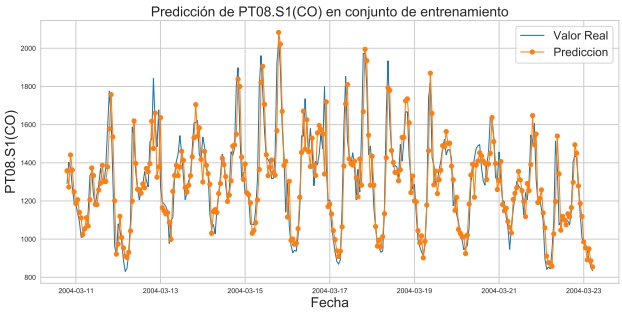

Figura 4.1. Ajustes del modelo ARIMA en el conjunto de entrenamiento

La predicción en el conjunto de test se muestra en la Figura 4.2 donde solo se muestran los primeros 100 datos. Como se puede observar la predicción presenta un leve desfase con respecto al valor real. De todas maneras la predicción preserva la tendencia de los datos.

Sumado a lo anterior se estudia el rendimiento del modelo para predecir datos a más pasos en el futuro ya que es de interés poder prever la concentración del contaminante en el aire para el día siguiente. En la Figura 4.3 se muestra la predicción a 24 horas a futuro donde cada color representa un bloque distinto.

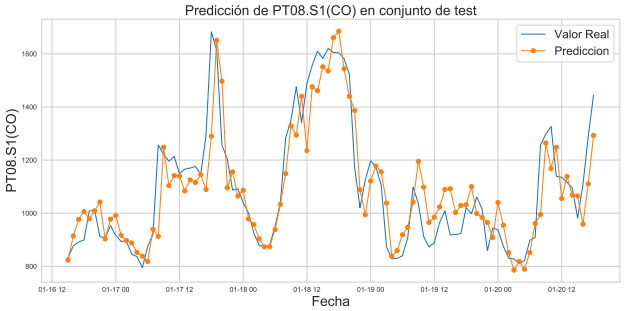

Figura 4.2. Ajustes del modelo ARIMA en el conjunto de test

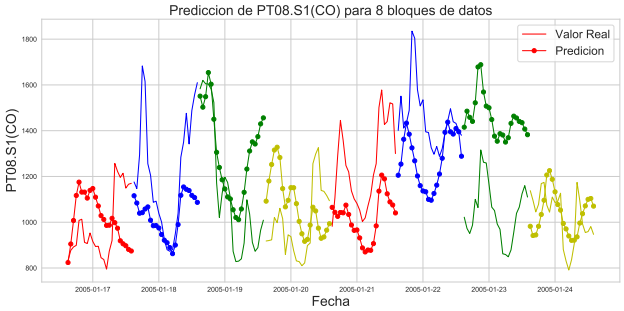

Figura 4.3. Predicciones para pasos de 24 horas a futuro

Como se puede observar los primeros datos tienden a estar cercanos a la curva pero posteriormente, al acercarse al paso 24 horas, la predicción se aleja del valor real. Además, para solo un par de bloques la predicción logra capturar la tendencia de los datos.

Para cuantificar el rendimiento en los respectivos conjuntos se estudia el Error Absoluto Medio (MAE) y el Error Cuadrático Medio (MSE) los cuales se muestran en la Tabla 4.1. Se tiene que para el conjunto de test el error es menor comparado con el del conjunto de entrenamiento, lo que puede considerarse como que el modelo tiene buena capacidad de generalización, aunque gráficamente se tiene un leve desfase. Para los errores con paso de 24 horas se tiene que estos aumentan considerablemente lo cual era de esperarse ya que es muy difícil poder predecir tan a futuro variables de este estilo.

	MAE	MSE
Error Entrenamiento	61.988	7463.359
Error Test	56.662	6517.501
Error Test (paso 24h)	121.253	27135.618

Tabla 4.1. Errores obtenidos para CO en los diferentes conjuntos

5. Redes Neuronales Recurrentes

5.1. Marco Teórico

Long-Short Term Memory (LSTM) es una arquitectura de red neuronal recurrente (RNN) ampliamente utilizada en el procesamiento de datos secuenciales, por ejemplo, para predicciones en series de tiempo. Esto último se debe a la capacidad que tienen las RNNs para generar feedback. En particular, LSTM tiene un lazo adicional que permite condicionar salidas futuras a entradas anteriores.

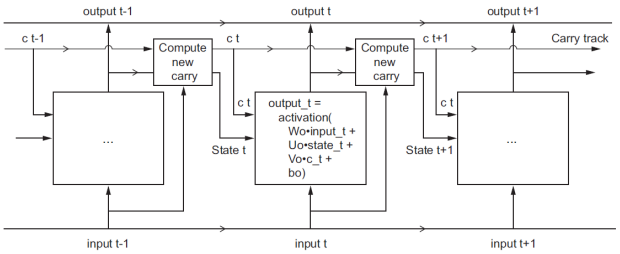

Figura 5.1. Estructura de LSTM.

En la Figura 5.1 se ve la arquitectura de una red LSTM, los lazos de retroalimentación de la red y su influencia en el cálculo de la función de activación. Se puede entender el flujo de datos del *carry* como una forma de modular el siguiente estado y salida.

5.2. Resultados y Análisis

Utilizando `TensorFlow`, la librería abierta creada por *Google* para aprendizaje automático, se diseñó una red compuesta por 5 capas: la capa de entrada, tres capas internas (ocultas) y la capa de salida. Las dos primeras capas ocultas de la red corresponden a dos modelos LSTM con las funciones de activación sigmoide y unidad rectificadora lineal (`ReLU`), respectivamente. Estas se encargarán de focalizar la importancia de cada variable y el "arrastre" de los tiempos anteriores en el modelo. La última capa oculta y la capa de salida serán capas simples con la función de activación `ReLU`, estas serán las encargadas de generar la regresión asignando los pesos finales para proporcionar el `output`. Cabe destacar que, en pos de evitar la linealidad del modelo, se consideraron las variables de la base de datos, los 24 datos de horas anteriores de CO y una base cuadrática de estos mismos valores. Es decir, cada valor al cuadrado como una entrada adicional. Además, como preparación previa, los datos fueron centrados en 0 y reescalados para obtener varianza 1, esto favorece el buen funcionamiento de las funciones predeterminadas de la librería (para comparar los modelos se invierte tal escalamiento).

La Figura 5.2 muestra la comparación de 120 datos del conjunto de testeo con sus respectivas predicciones para el modelo de regresión considerando una hora de proyección futura. El valor MSE obtenido en este conjunto fue 4721.9, mientras que el valor MAE fue de 49.85, con lo que desde un principio podemos hacer notar una diferencia respecto al modelo ARIMA. Este hecho será discutido con profundidad más adelante. De la misma forma, para el modelo proyectado a 24 horas a futuro se logró un MSE de 18248.67 y un MAE de 104.63. En la Figura 5.3 se muestra la estimación para cuatro bloques de 24 horas en contraste con sus valores reales.

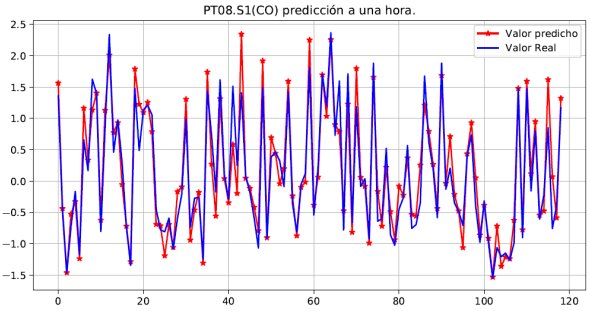

Figura 5.2. Comparación valor real y predicho para proyección a una hora con LSTM, conjunto test. El nivel de ajuste es tal que ambas curvas tienden a superponerse.

Además de estos dos modelos, se probaron dos modelos extras que buscaban la predicción de los mismos valores pero en otros niveles de alcance temporal; 6 horas y 10 horas. El detalle será omitido en este trabajo pero es importante mencionar que las funciones de pérdida (MSE y MAE) aumentan proporcionalmente al aumento de la dimensión buscada para el `output`, evidenciando el *trade-off* existente entre el alcance de la predicción y el nivel de precisión en la estimación.

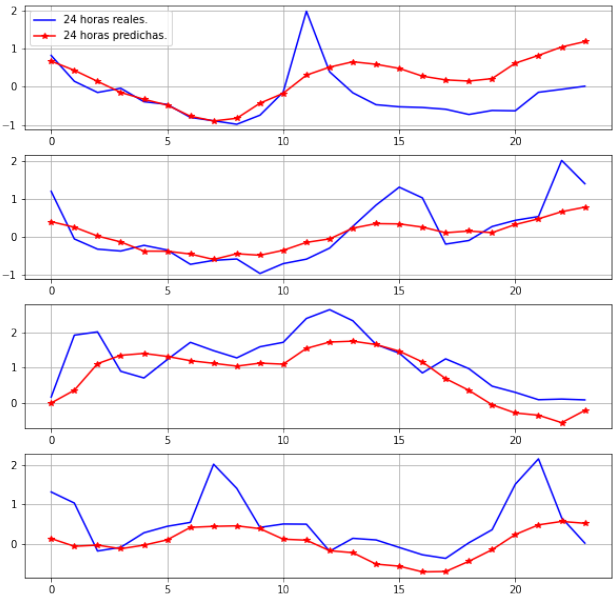

Figura 5.3. Cuatro muestras de 24 horas de estimación con el modelo de red neuronal (conjunto testeo).

Un aspecto interesante de los resultados recién mencionados en la Tabla 5.1 es que pareciese que el ajuste en el conjunto

	MAE	MSE
Entrenamiento 1 hr.	56.78	5938.2
Test 1 hr.	49.85	4721.9
Entrenamiento 24 hrs.	107.144	20700.27
Test 24 hrs.	104.63	18248.67

Tabla 5.1. Resumen errores para modelos RNN's.

de testeo es siempre mejor que en el conjunto de entrenamiento. Esto tiene dos explicaciones, la primera y más obvia es debido a la diferencia de tamaño de estos conjuntos. Si tengo más observaciones tengo más de dónde acumular error. La segunda razón es debido al algoritmo de entrenamiento de la red, donde el 80% de los datos considerados como conjunto para entrenar son, de hecho, sólo 70% y el 10% restante corresponde al *conjunto de validación*. De esta forma, en cada iteración, el algoritmo ajusta los parámetros para estimar el conjunto de entrenamiento, pero sólo conserva aquellas iteraciones donde el modelo minimice los errores para el conjunto de validación. Así podríamos decir que el algoritmo *prioriza evitar el sobreajuste*, generando un modelo consistente a cualquier conjunto de observaciones.

También podemos comentar que los modelos de RNN son extremadamente sensibles a los hiperparámetros necesarios en su arquitectura. Por ejemplo; cantidad de capas, cantidad de nodos en cada capa, método de optimización, *learning rate*, *epochs* y la aleatoriedad de los pesos iniciales de la red antes del entrenamiento (punto inicial del algoritmo). De esta forma, aún obteniendo errores menores es necesario recordar que existe una variabilidad no despreciable en estos resultados.

6. Comparación de modelos

La Tabla 6.1 muestra los valores de los errores obtenidos en cada modelo para el conjunto de testeo. Esto nos genera como primera impresión, que el modelo de redes neuronales posee un poder de estimación más alto en ambos casos (1 hora y 24 horas de estimación). Por lo mencionado en los párrafos anteriores, además sabemos que un 10% de los datos correspondieron a información no utilizada al momento de entrenar la red neuronal, en este caso entrenar con la totalidad del conjunto de entrenamiento jugó en contra para el modelo ARIMA.

	MAE	MSE
ARIMA 1 hr.	56.662	6517.501
RNN 1 hr.	49.85	4721.9
ARIMA 24 hr.	121.253	27135.618
RNN 24 hr.	104.63	18248.67

Tabla 6.1. Resumen comparación de errores en modelos ARIMA y RNN.

Sin embargo, replicar el algoritmo con el que se construyó la red neuronal no nos asegura volver a obtener un resultado mejor que los modelos ARIMA, existe la probabilidad no nula de ser necesario entrenar el modelo en más de una ocasión desde cero para generar una red aceptable.

Además de lo ya dicho, cabe mencionar que el modelo de redes neuronales es mucho más pesado (en cuanto a unidad de información en computación) por el tipo de entrenamiento y el uso de la base cuadrática de los datos que duplica la cantidad de columnas en nuestro *dataset* (y no precisamente la cantidad de información).

7. Conclusión

Es poco creíble que para cualquier serie de tiempo, el modelo que mejor describa el comportamiento de las variables sea uno basado en relaciones lineales entre los datos como lo hace los modelos ARIMA. Sin embargo, no deja de ser increíble el nivel de aproximación que puede llegar a tener para un método así de veloz y replicable al momento de entrenar. Esto lo dota de una gran importancia en la práctica, en ocasiones donde tener una estimación futura dentro de un rango sea de más utilidad que un valor cerrado, como por ejemplo procesos controlados por bandas de control y metodologías 3 sigmas o 6 sigmas.

Pero si de precisión se habla, la innovación de las redes neuronales y sus versiones recurrentes dan respuestas claras y ajustes certeros. Sin embargo, la arquitectura de las redes neuronales aún no posee una metodología eficiente para ajustar sus hiperparámetros, por lo que es difícil evitar el "ensayo y error" al momento de la creación de modelos de este tipo. Esta misma razón nos dice que la red encontrada en este trabajo tampoco es la óptima que se pueda lograr para este conjunto de datos, es posible que otra configuración de capas y nodos produzca estimaciones aún mejores.

Entre más complejo sea el caso, y más compleja sea su respuesta, también serán mayores los requerimientos computacionales necesarios para producir estimaciones. Algunas rutinas de entrenamiento de las redes ocupadas en este trabajo demoraron entre 1 a 3 horas en su ejecución por falta de un equipo eficiente. Pese a esto, el uso de estas herramientas cada vez está más al alcance de cualquier persona gracias a librerías documentadas que se actualizan y mejoran constantemente como los son `TensorFlow` o `PyTorch`, y eso motiva aún más su estudio.

Bibliografía

[1] BROWNLEE, JASON. *Deep Learning for Time Series Forecasting: Predict the future with MLPs, CNNs and LSTMs in Python.* Edición v1.4, 2014.

[2] CHOLLET, FRANÇOIS. *Deep Learning whith Python.* 2018.

[3] `www.keras.io/api/` , *Keras API References.*

[4] `www.tensorflow.org/api_docs/python/`, *TensorFlow: API Documentation.*

[5] `https://bit.ly/3h1iIn6`, *UCI Machine Learning Repository: Air Quality Data Set.*