

INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

ITAM

**Diseño de una red neuronal para pronóstico de demanda en
un sistema de bicicletas compartidas**

Tesina que para obtener el título de :

Ingeniero Industrial

Presenta:

Hugo Villegas González

Índice

1. Sistemas de bicicletas compartidas en el mundo	1
1.1. Problemas de los BSS	2
1.2. Ecobici	4
2. Agrupamiento	12
2.1. K Means	13
2.2. Affinity Propagation	15
3. Redes neuronales	17
3.1. Algoritmo de Backpropagation	19
3.2. Recurrent Neural Networks	22
3.3. LSTM	24
3.4. Implementación de una LSTM para predicción de demanda	25
4. Predicción de demanda para los clusters	29
5. Conclusión y posibles mejoras	31

Índice de figuras

1.1. Viajes por semana de Ecobici	5
1.2. Estaciones con mayor actividad durante los últimos 3 años	5
1.3. Suma de llegadas y salidas en el 2016 por estación	6
1.4. Diferencia entre salidas y llegadas en el 2016 por estación	6
1.5. Mapa de calor por llegadas y salidas de los últimos 3 años	7
1.6. Viajes al año entre 2 estaciones	8
1.7. Viajes por día de la semana	9
1.8. Número de viajes en días laborales del 2016	10
1.9. Flujo neto (llegadas - salidas) para diferentes martes en la estación 27 .	11
2.1. Datos por agrupar	12
2.2. Suma de cuadrados para k clusters con el codo indicado	14
2.3. Agrupación por k means usando k=10	14
2.4. Ilustración de como gradualmente affinity propagation converge a un resultado. Tomada del paper de Brendan (2017)	15
2.5. 22 clusters generados con affinity propagation	16
2.6. 14 clusters generados con k means	16
3.1. Estructura de una red nueronal simple	18
3.2. Nomenclatura básica de una red neuronal	20
3.3. Estructura básica de una Red Neuronal Recurrente. Tomada de Servan- Schreiber et al. (1991)	22

3.4. Algoritmo de BPTT. Tomada de Backpropagation through time - Wikipedia	23
3.5. Efectos de aplicar una función sigmoide múltiples veces. Tomada de Recurrent neural network - Wikipedia	24
3.6. Representación de una LSTM. Tomada de Understanding LSTM Networks - Colah's blog	25
3.7. Representación del modelo de la red neuronal	26
3.8. Red con 30 epochs de entrenamiento prediciendo demanda para todo el sistema	27
3.9. Red con 50 epochs de entrenamiento prediciendo demanda para todo el sistema	27
3.10. Red con 200 epochs de entrenamiento prediciendo demanda para todo el sistema	27
3.11. Predicción de casi un día con retroalimentación de la red	28
3.12. Predicción usando el modelo de todo el sistema para pronóstico de un cluster	28
4.1. Redes neuronales del cluster 8 con diferentes epochs	29
4.2. Predicción de próximas 5 horas para momento aleatorio del cluster 8 .	30

Índice de tablas

1.1. Viajes más comunes durante 2016	8
1.2. Viajes por día de la semana	9
4.1. Métricas para diferentes epochs	29

1. Sistemas de bicicletas compartidas en el mundo

El transporte es el mayor problema de infraestructura en las megaciudades, no solo por la dificultad de mover a millones de personas sino que también generan problemas de contaminación y congestión que se reflejan en toda la zona urbana [Hazel y Miller, 2007].

Las ciudades están implementando diversas alternativas de transporte para afrontar estos problemas, y el que tiene mayor tasa de expansión es el sistema de bicicletas compartidas (BSS por sus siglas en inglés). Esto quizá se debe a que la bicicleta ha estado presente en la sociedad por más de un siglo, con un diseño que ha cambiado poco debido a su sencillez y eficiencia para transportarse. Adicionalmente es un medio de transporte limpio y económico, por lo que resulta vital para las megaciudades promover el uso de estas.

Los BSS como los conocemos se implementaron con éxito hasta la década de los 90s en ciudades europeas, y apartir de entonces se han extendido al resto del mundo con un crecimiento acelerado. A final del año 2015 existían alrededor de 1,270,000 bicicletas para este tipo de sistemas, cifra que aumentó a 2,294,000 para finales del 2016, mismas que se encuentran repartidas en 1175 ciudades de 63 países.[The bike sharing blog, 2017].

1.1. Problemas de los BSS

Los mayores problemas logísticos que tienen estos sistemas son el rebalanceo y el enrutamiento de los vehículos [Schuijbroek et. al., 2013]. Es necesario identificar los niveles deseados de inventario por estación a lo largo del día para brindar un nivel de servicio alto a los usuarios. Por ejemplo: un usuario podría querer estacionar una bicicleta en la estación más cercana a su destino, pero al darse cuenta que no hay espacios disponibles debe recorrer mayor distancia para estacionarse y luego caminar, lo mismo sucedería para querer tomar una bicicleta de una estación vacía.

Por la naturaleza del sistema, hay variaciones que provocan saturación y escasez en las estaciones. Se necesita un rebalanceo para satisfacer la demanda continua de bicicletas y esto se logra a base de vehículos que muevan bicicletas de unas estaciones a otras; esto implica costos de gasolina, sueldos y tiempo. Por lo tanto se necesita tener una buena predicción de la demanda y posteriormente hacer una buena planeación de la(s) ruta(s) para el rebalanceo y reducir los gastos.

Se han hecho investigaciones para poder tener una mejor predicción de la demanda. Algunos de los sistemas de grandes ciudades toman suposiciones e hipótesis sobre variables demográficas, uso de tierra, y factores económicos y de infraestructura para realizar pronósticos de demanda [Ahmadreza et. al., 2017]. Ahmadreza sugiere que estas variables afectan de manera distinta la demanda que el rebalanceo, ya que con mayor número de estaciones, incluso siendo de menor capacidad provocan un autobalanceo por parte de los usuarios.

También se ha descubierto que el clima impacta la demanda de los viajes en bicicleta, reduciéndolos cuando hay lluvia, frío, y niveles altos de humedad. Las estaciones cercanas a entradas del Metro se ven más afectadas durante estas condiciones que las que se encuentran lejos del Metro. Pero se ven poco afectadas en horas de poca luz [Gebhart y Noland, 2017].

También se pueden hacer clusters de estaciones para facilitar la predicción. En el paper publicado por Li Y. titulado *Traffic Prediction in a Bike-Sharing System* se hace un modelo para predicción de la demanda tomando en cuenta clusters de estaciones, patrones de cambio históricos y el clima. Los clusters se forman de acuerdo a la ubicación y comportamientos en la demanda de las estaciones. Con esto deja de ser relevante analizar las estaciones de manera individual y se puede hacer un pronóstico de demanda para cada clúster en intervalos de tiempo con resultados que se apegan a la realidad tanto en días normales como con anomalías. A mayor número de clusters se pierde precisión. *Optimal inventory management of a bike-sharing station*, paper publicado por Raviv T. y Kolka O. en el 2013 menciona que modelando la insatisfacción de los usuarios como costo monetario y sumando el costo de rebalancear, se puede modelar un inventario inicial óptimo para un intervalo de tiempo T. También menciona que es necesario un almacén no disponible al público del que se puedan tomar o guardar bicicletas, para ayudar en el rebalanceo.

Actualmente la mayoría de los BSS tienen sistemas en tiempo real para conocer el inventario, pero aun así es necesario tener una predicción de la demanda ya que toma tiempo realizar el rebalanceo. Esta demanda puede cambiar a lo largo del día por factores como el clima, o de una semana a otra (la demanda de un Lunes puede ser diferente al Lunes

anterior por diferentes razones; fue día feriado; hay un desfile que bloquea las calles principales de la ciudad; algunas empresas decidieron dar asueto; etc.). Adicionalmente hay una correlación entre las demandas de estaciones por lo que hacer un pronóstico de la demanda resulta ser una tarea complicada.

1.2. Ecobici

México tiene 5 sistemas de bicicletas compartidas en funcionamiento; se encuentran en las ciudades de Toluca, Guadalajara, Ciudad de México, Hidalgo y Puebla (consultado en *The bike sharing world map* Julio de 2017), de los cuales Ecobici, en la Ciudad de México es el más grande del país con 452 estaciones y más de 250,000 usuarios registrados (información consultada en *The bike sharing world map* y *Estadísticas Ecobici* el 7 de Julio de 2017).

Los problemas logísticos para Ecobici son mayores, ya que es un sistema muy grande (entre los 25 más grandes del mundo por número de estaciones y número de bicicletas) y el tráfico constante en la ciudad hace más complicado el rebalanceo. Es necesario realizar un análisis detallado del comportamiento que tiene el sistema para proponer alternativas eficientes a sus problemas logísticos.

Desde que inició operaciones en febrero del 2010 Ecobici no ha dejado de crecer de alguna manera u otra, pero su crecimiento no ha sido constante. En las últimas 2 semanas de fin de año y semana santa los viajes se reducen drásticamente, el resto de las semanas parece tener una variabilidad aleatoria. Observando la gráfica uno puede darse cuenta que tener un pronóstico acertado de la demana es tarea difícil. Incluso parece

ser que el número de viajes se ha estancado desde el 2015



Figura 1.1: Viajes por semana de Ecobici

Por otro lado, las estaciones parecen tener un comportamiento definido. Durante los últimos 3 años, la estación 27 Reforma Havre ha sido la mayor demandada, superando por mucho al resto de estaciones en número de arribos y salidas.

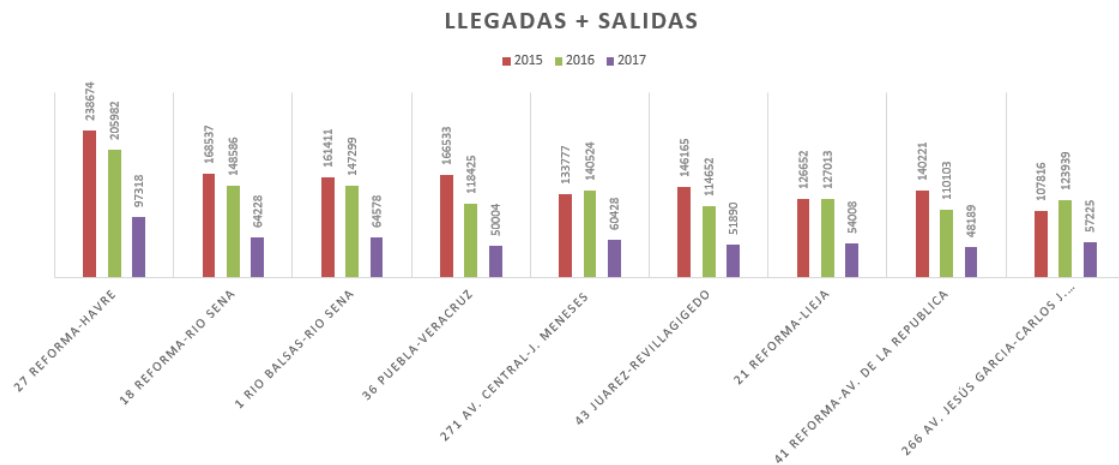


Figura 1.2: Estaciones con mayor actividad durante los últimos 3 años

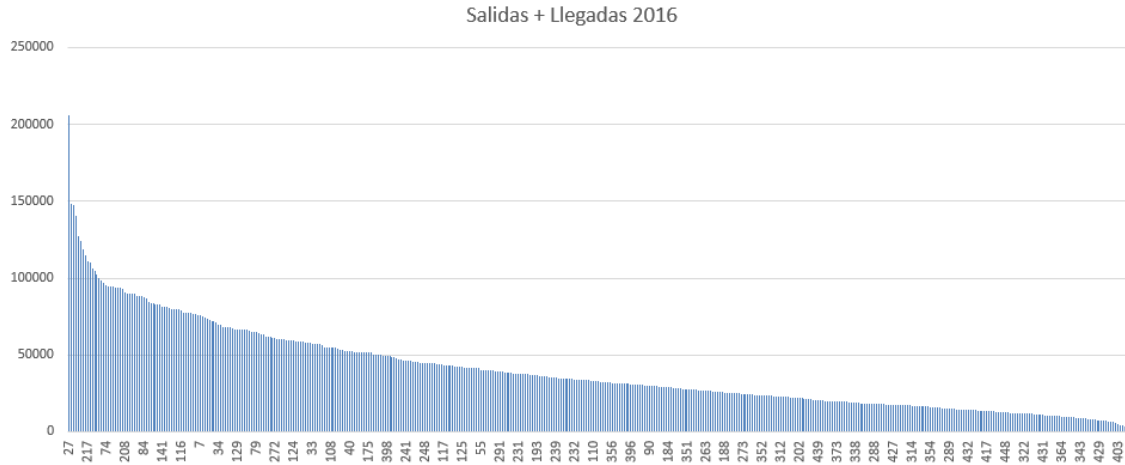


Figura 1.3: Suma de llegadas y salidas en el 2016 por estación

En cambio, si se hace la diferencia entre salidas y llegadas se puede ver que hay estaciones que naturalmente tienen mayor número de salidas que llegadas, otras con más llegadas y un tipo de estación que parece tener un balance entre llegadas y salidas.

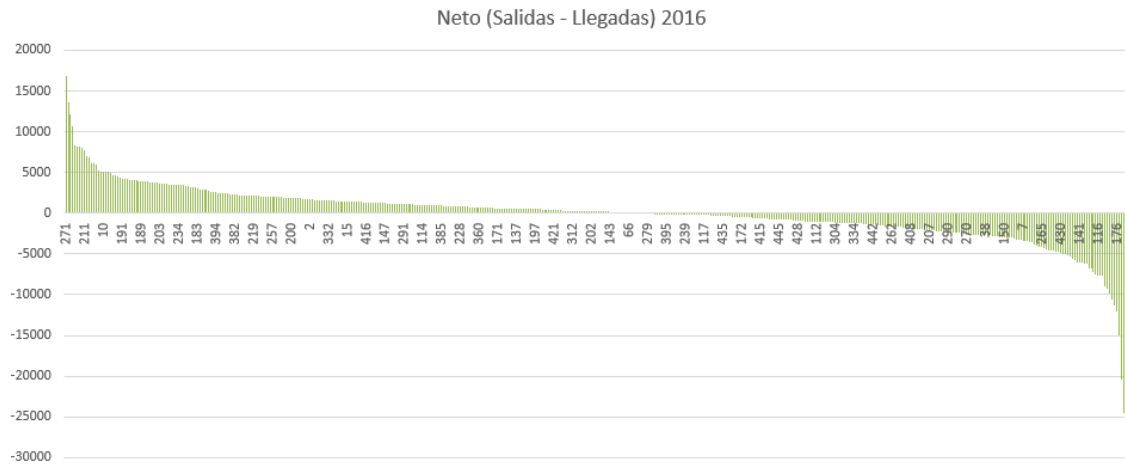


Figura 1.4: Diferencia entre salidas y llegadas en el 2016 por estación

Aunque esto puede generar una interpretación errónea si no se analiza a mayor profundidad como se hará mas adelante. Por ejemplo las estaciones *266 Av. Jesús García-Carlos J. Meneses* y *267 AV. Jesús García-Carlos J. Meneses* son las que tienen mayor número

de llegadas, pero son contiguas a las estaciones *271 AV. Central-J. Meneses* y *272 AV. Central-J. Meneses* las cuales son 1er y 6to lugar en número de salidas respectivamente. Como este hay más casos de estaciones contiguas con flujos recíprocos. La mayor actividad dentro de la Ciudad de México se genera en avenida Reforma y la zona de Buenavista. Después le siguen las colonias Condesa y Roma Norte.

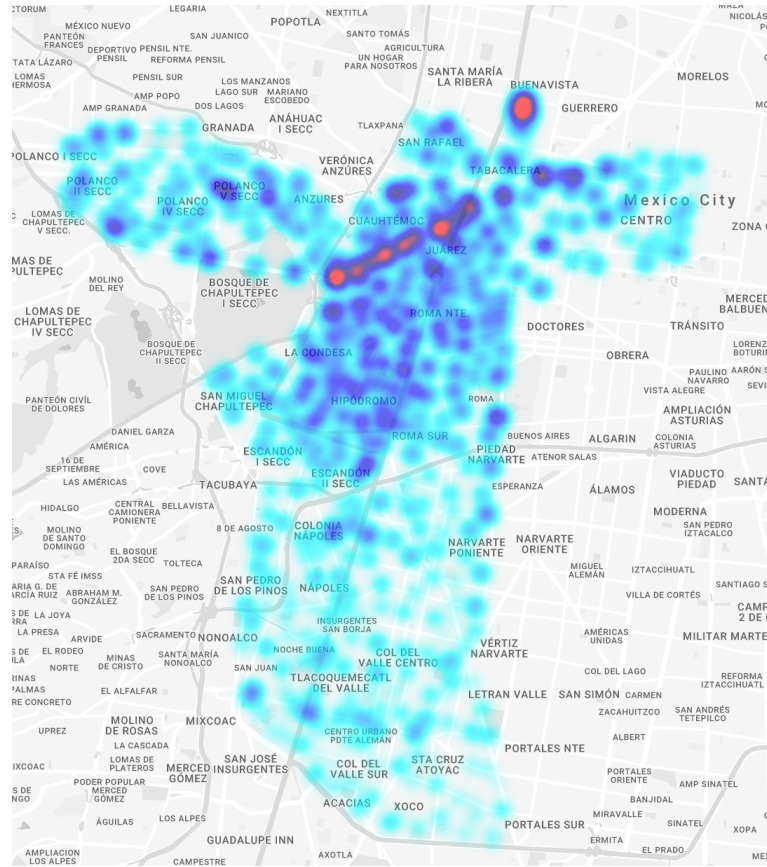


Figura 1.5: Mapa de calor por llegadas y salidas de los últimos 3 años

Pero los viajes más frecuentes entre 2 estaciones no son en ninguna de las zonas mencionadas. Los más comunes se dan entre la estación *211 Newton-Horacio* y *217 Euler-Av. Horacio* ubicadas en Polanco con poco más de 2300 viajes durante el 2016.

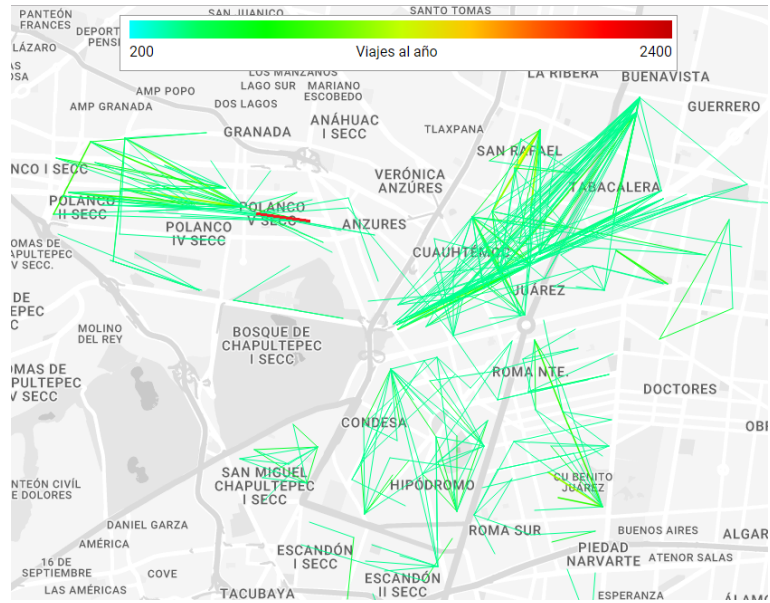


Figura 1.6: Viajes al año entre 2 estaciones

Estaciones	Total de viajes
211 - 217	2321
174-183	1631
1-18	1536
85-174	1244
174-257	1216
206-208	1204
21-27	1158

Tabla 1.1: Viajes más comunes durante 2016

Parece ser que hay 2 comportamientos en los viajes realizados durante diferentes días de la semana. De Lunes a Viernes el número de viajes totales es parecido, pero disminuye durante los fines de semana. En los últimos 3 años el día martes ha sido el que más viajes ha tenido y domingo el de menor actividad.

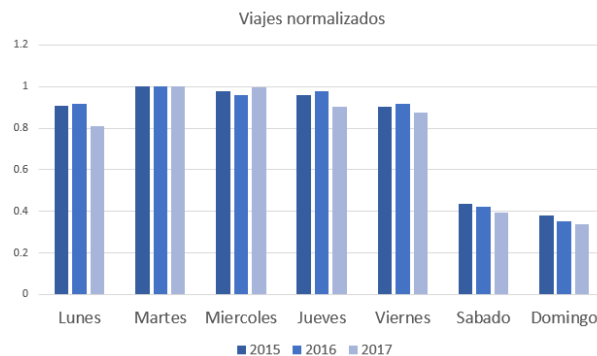


Figura 1.7: Viajes por día de la semana

	Actividades por año		
	2015	2016	2017
Lunes	1500917	1501661	619677
Martes	1651031	1636017	764333
Miercoles	1613269	1572465	760441
Jueves	1586421	1598553	689980
Viernes	1493699	1498526	667820
Sabado	716738	689625	300114
Domingo	629794	572640	258865

Tabla 1.2: Viajes por día de la semana

De ahora en adelante se utilizarán solamente 1 conjunto de datos acotado en base a 2 supuestos.

1. Solo se utilizarán los días entre semana ya que son los relevantes para análisis y estudio del sistema.
2. Se trabajarán solo los años 2015 y 2016 para análisis y estudio ya que la actividad en ambos fue parecida. El año 2017 se utilizará para prueba de hipótesis.

Todos los días entre semana parecen comportarse de manera parecida; hay un pico de actividad alrededor de las 9:00, disminuye y vuelve a aumentar un poco a las 14:00, finalmente a las 19:00 hay otro pico parecido al de la mañana pero con una disminución gradual. Coincide con el comportamiento de un día laboral: la gente se va a trabajar en la mañana y algunos salen a comer, a las 7 ya es hora de salida. Los días viernes hay menor actividad después de las 5 de la tarde que los otros días.

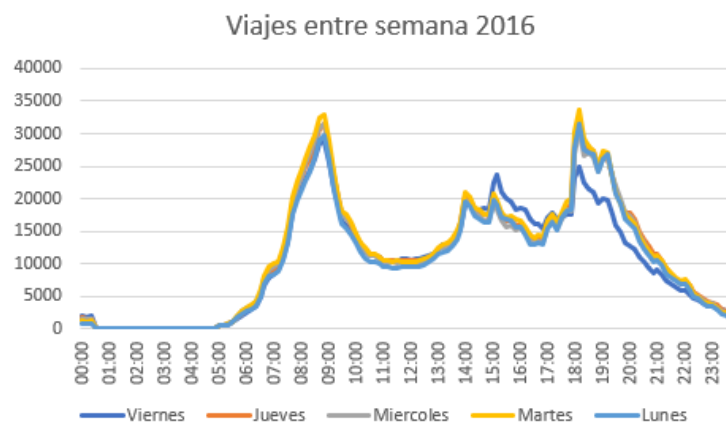


Figura 1.8: Número de viajes en días laborales del 2016

Algo que podría complicar aún más el rebalanceo es que los cambios de una semana a otra son mayores a nivel de estación, por ejemplo la estación 27 durante los martes parece tener un flujo controlado (con poca variación) durante las primeras horas del día, pero después de la 1 de la tarde este flujo cambia mucho de un martes a otro. Esto puede deberse a fallos en el rebalanceo, cambios en el clima, comportamiento anormal de la gente o algún otro factor que cambie la demanda de bicicletas para cierta estación.

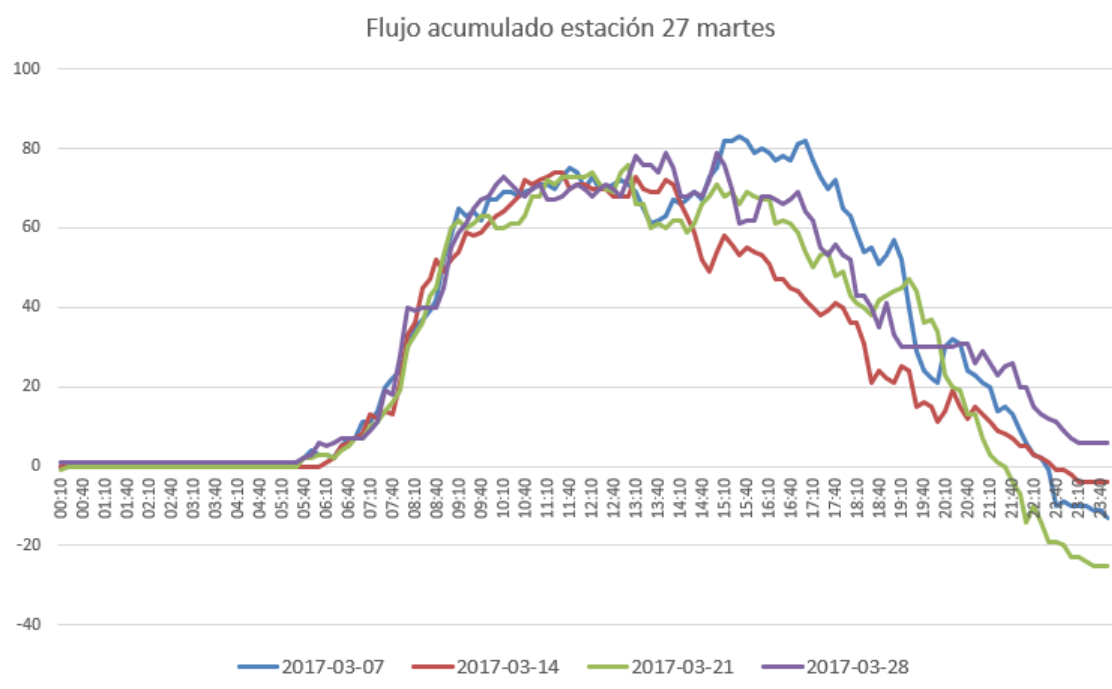


Figura 1.9: Flujo neto (llegadas - salidas) para diferentes martes en la estación 27

2. Agrupamiento

El agrupamiento, también conocido como *clustering*, es una técnica que consiste en agrupar elementos dependiendo de sus similitudes, el objetivo es que los datos dentro de un grupo sean similares entre sí y diferentes de los datos de otros grupos (Tan et. al., 2006). Así resultará más fácil analizar grupos de estaciones en lugar de las 452 estaciones por separado.

Vogel (2011) sugiere utilizar la actividad por estación para realizar el agrupamiento, en este caso se utilizará el número total de viajes en el año 2016 excluyendo fines de semana. A esta sugerencia se añaden las coordenadas de la estación, ya que para objetivos de este trabajo es necesario que las estaciones se encuentren cerca. Los datos pueden visualizarse en las siguientes figuras.

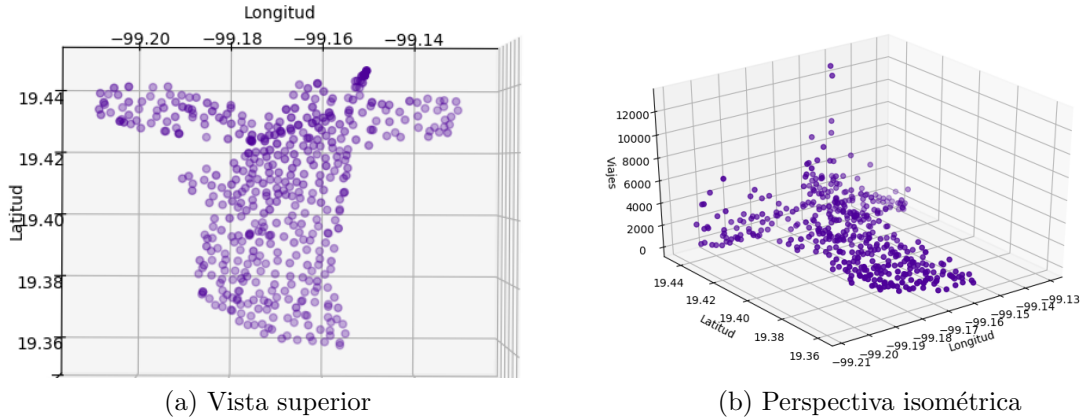


Figura 2.1: Datos por agrupar

Tan et. al. (2006) sugiere comparar los resultados de diversos métodos de agrupamiento para diferentes números de clusters.

2.1. K Means

Algoritmo usado para dividir n observaciones en k grupos de manera que cada grupo tenga una varianza mínima entre sus observaciones. El algoritmo consta de los siguientes pasos:

1. **Asignación.** Cada dato es asignado al centroide más cercano de acuerdo con la distancia euclidiana.
2. **Actualización.** Los centroides son recalculados de usando el promedio de sus datos respectivos.

El algoritmo itera en estos dos pasos hasta que se llega al criterio de finalización (i.e., ningún dato cambia de grupo, la suma de distancias se ha minimizado, número máximo de iteraciones).

No hay un algoritmo universal para escoger k , pero hay varios métodos que sugieren k , uno de ellos es el criterio del codo (*elbow method* en inglés). Como Kassambara (2017) sugiere, la suma de cuadrados de los cluster debe ser mínima, pero hay un punto en el que el cambio comienza a ser mínimo, formando un “codo” si se grafica la suma de cuadrados para n clusters. A partir de este punto el impacto de aumentar k disminuye considerablemente.

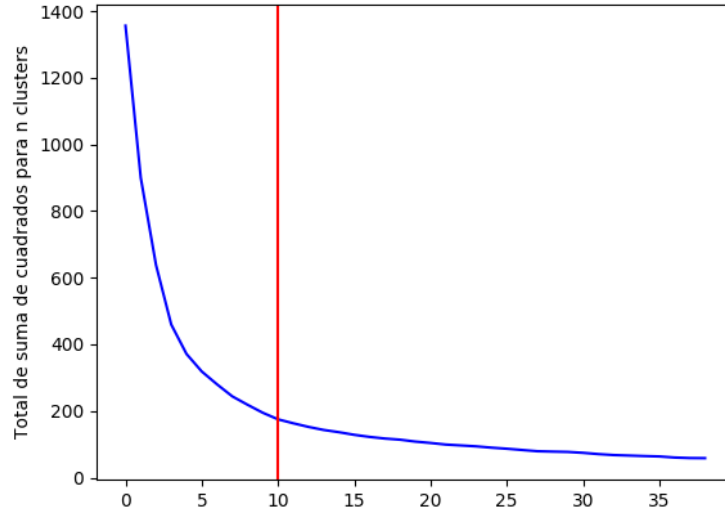
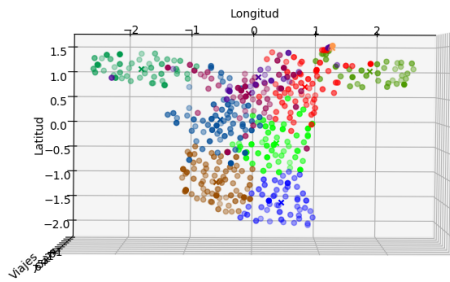
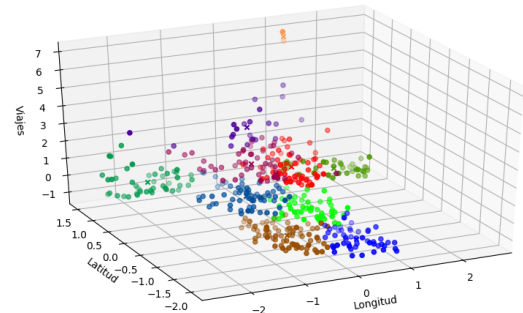


Figura 2.2: Suma de cuadrados para k clusters con el codo indicado

Este criterio arroja como resultado $k = 10$ y los clusters se ven de la siguiente manera:



(a) Vista superior



(b) Perspectiva isométrica

Figura 2.3: Agrupación por k means usando $k=10$

Es importante recalcar que $k = 10$ es un punto subjetivo a partir del cual el cambio en la suma de cuadrados total es mínima, esto no significa que una elección de k mayor a 10 sea mala.

2.2. Affinity Propagation

Es otro método de agrupación que funciona con base en un intercambio de mensajes entre los puntos (Brendan, 2017). Este método no necesita que sea indicado el número de clusters a diferencia de **K means**.

El algoritmo consiste en ver todos los puntos como una red, y transmitir mensajes valorados a través de ella. Los mensajes son actualizados con base en fórmulas que buscan el mínimo de una función de energía apropiada. El mensaje representa la afinidad que tiene un punto para relacionarse con otro. Hay 2 funciones para representar esto; la función de “responsabilidad” que refleja la evidencia acumulada para indicar que tan adecuado es el punto i con k ; la otra función es de “disponibilidad” que representa que tan apropiado es para el punto k relacionarse con i de acuerdo con la evidencia acumulada que ha tenido k con otros puntos. Se generan iteraciones hasta que converge a un resultado.

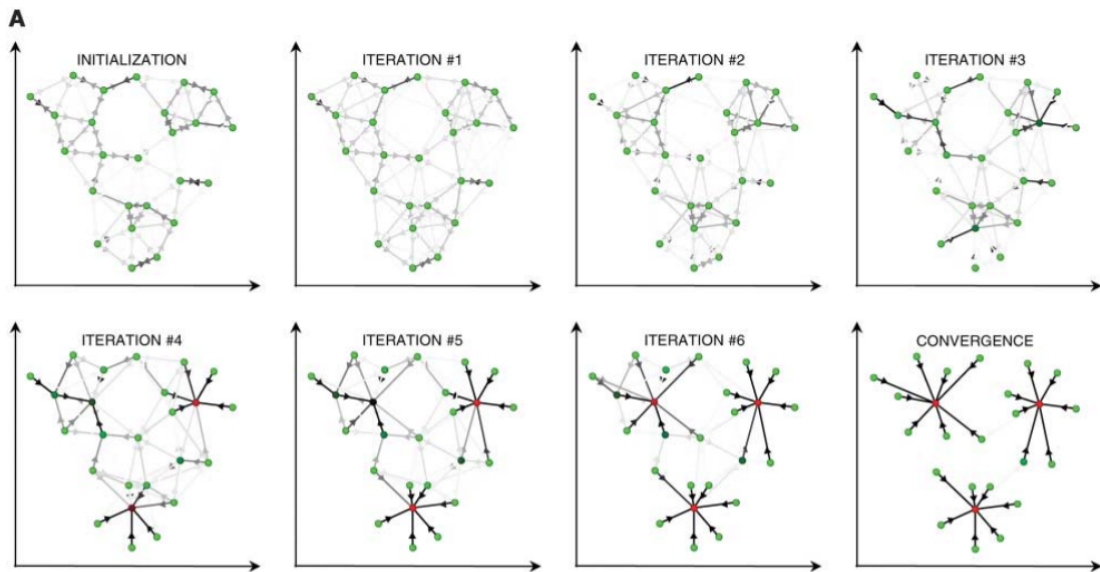


Figura 2.4: Ilustración de como gradualmente affinity propagation converge a un resultado. Tomada del paper de Brendan (2017)

Este algoritmo indica que son 22 grupos, los resultados se indican en la siguiente figura

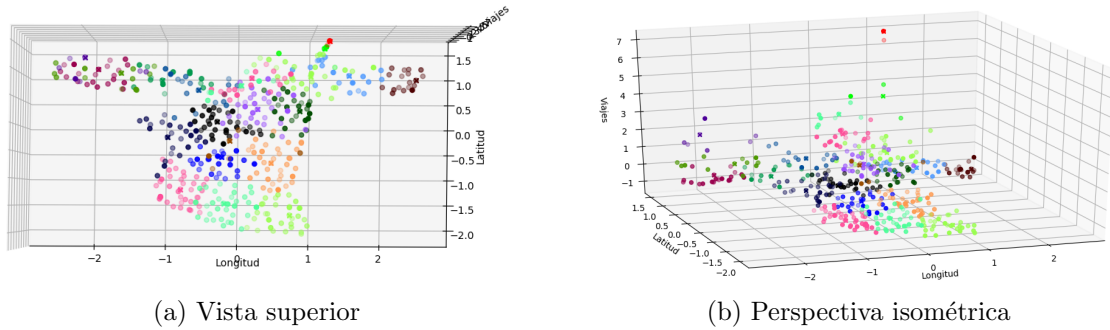


Figura 2.5: 22 clusters generados con affinity propagation

Como era de esperarse los resultados difieren del algoritmo de k means, incluso si se utiliza $k = 22$ los resultados no son iguales. Pero esto sugiere que puede usarse un número mayor de clusters, por lo que se aumentará a 14 el valor de k para generar los clusters para usar en lo que resta de este trabajo.

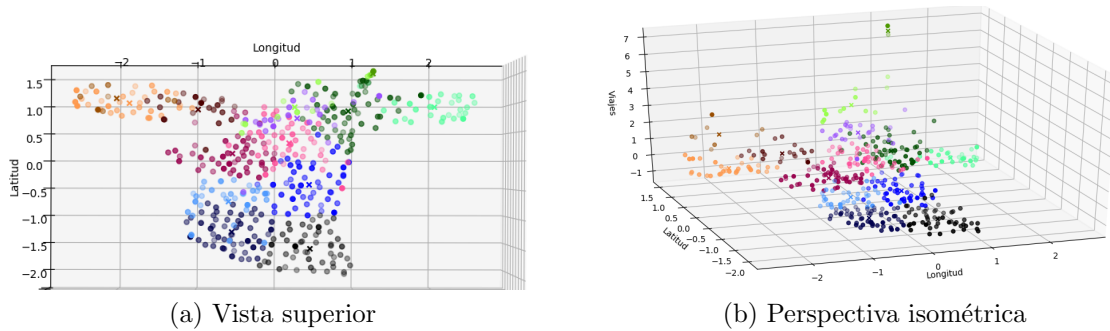


Figura 2.6: 14 clusters generados con k means

3. Redes neuronales

Como se mencionó en la primer sección de este documento, es necesario tener una buena predicción sobre el tráfico en el sistema. Algunos autores han propuesto utilizar redes neuronales para modelar estos problemas [Zhao et. al, 2017]. Pero antes es necesario explicar qué es una red neuronal y porqué es factible resolver este tipo de problemas usándolas.

Las redes neuronales son sistemas computacionales diseñados para que por medio de un entrenamiento con base en valores de entrada y sus respectivos valores de salida sean capaces de “aprender” una función $y = f(x)$ [Nielsen, 2015]. Para el entrenamiento se necesita definir *epochs* y tamaño del *batch*.

Un *epoch* es recorrer toda la información de entrenamiento, así a mayor número de epochs se recorre más veces la información de entrenamiento. Un bajo número de epochs puede significar que la red no aprenda la función, por el contrario un número alto puede traer problemas de sobreajuste, siendo incapaz la red de reaccionar a información nueva.

Un *batch* o lote, es una parte de la información de entrenamiento. Entre más grande sea el lote es mayor la memoria computacional necesaria, un número bajo implica mayor número de iteraciones para completar un *epoch*. Por ejemplo si se tienen 1000 ejemplos de entrenamiento y un tamaño de lote de 500 se necesitan 2 iteraciones para completar un *epoch*.

Las redes neuronales están formadas por neuronas artificiales que simulan la conexión dentro un cerebro, donde las neuronas conectadas unas con otras generan enlaces que pueden incrementar o disminuir el estado de activación de las neuronas circundantes.

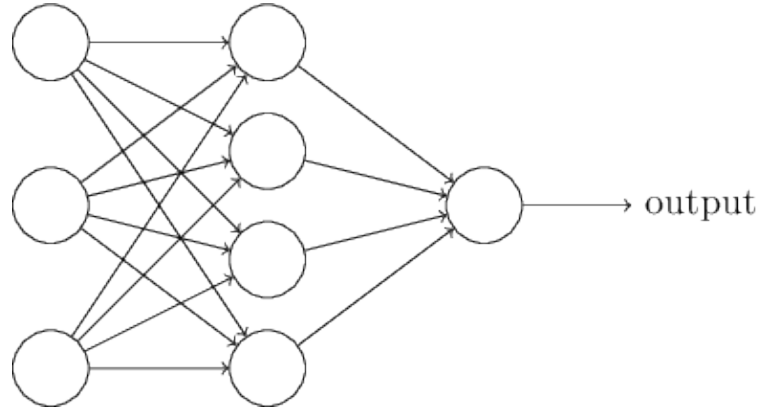


Figura 3.1: Estructura de una red neuronal simple

Como una neurona puede tener múltiples entradas, es necesario que tengan diferente importancia, para esto se multiplica por un valor w (por weight en inglés). Por otro lado cada neurona tiene un umbral de activación b (por bias en inglés). Típicamente la función de salida de la neurona es una función sigmoide, esto para poder representar pequeños cambios en las entradas de manera adecuada.

$$\sigma(z) = \sigma(wx + b) \quad (1)$$

Se necesitan encontrar valores para cada w y b tal que la salida de la red se aproxime a la función $y(x)$. Para medir que tan buena es esta aproximación se define una función de costo:

$$C(w, b) = \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

Donde w denota todos los *weights* en la red, b todos los *biases*, n es el número total de valores de entrada de entrenamiento, a es la salida de la red para cada entrada x . El $1/2$ se usa por conveniencia para que en la derivada cancele el 2. Cuando $C(w, b) \approx 0$

significa que la salida de la red a es aproximadamente igual a $y(x)$, por lo que se deben modificar w y b para minimizar C .

Para encontrar los valores de w y b que minimicen se utiliza un algoritmo llamada ***gradient descent***, que funciona calculando el gradiente ∇C para mover los valores proporcionalmente en la dirección negativa, así eventualmente se moverán hasta encontrarse en un punto muy cercano al mínimo de la función. Se define con la siguiente ecuación:

$$\nabla v = -\eta \nabla C$$

Donde v representa el vector de variables que definen C y η es un valor positivo preferentemente pequeño para que los cambios en v sean graduales, es conocido como ***learning rate***. Para realizar los cálculos se utiliza otro algoritmo llamado ***backpropagation***.

3.1. Algoritmo de Backpropagation

El objetivo de este algoritmo es calcular las derivadas parciales $\partial C / \partial t$ y $\partial C / \partial b$ de la función C respecto a cada w y b utilizando 4 ecuaciones. Para explicar con más detalle se usará la siguiente nomenclatura:

- b_j^l es el *bias* de la neurona j en la capa l .
- w_{jk}^l es el *weight* que va de la neurona j en la capa $l - 1$ hacia la neurona k en la capa l
- a_j^l es la salida o activación de la neurona j en la capa l .

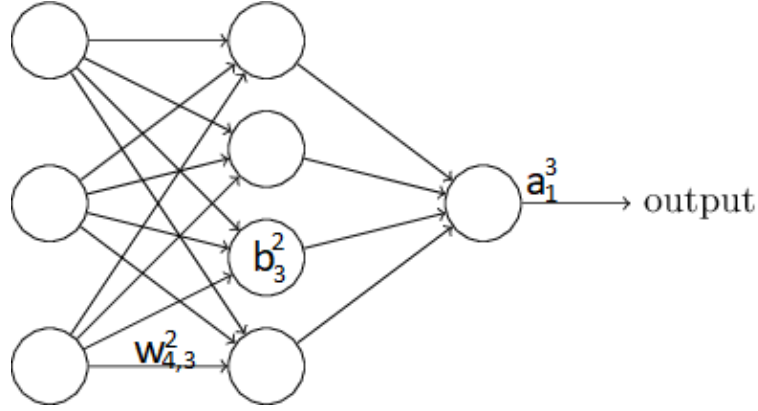


Figura 3.2: Nomenclatura básica de una red neuronal

Se necesita una ecuación para medir el error en cada capa:

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L) \quad (\text{BP1})$$

Donde δ_j^L es el error en la neurona j de la capa L , $\partial C / \partial a_j^L$ es el cambio de la función C respecto a la salida de la misma neurona, $\sigma'(z_j^L)$ es el cambio en la función respecto al vector de entrada z para la neurona j en la capa L . Es importante notar que en la ecuación (1) x puede ser sustituida por el vector de activación a de la capa anterior, ya que el objetivo de esta ecuación es obtener el error dado un vector de activación. La ecuación puede reescribirse de forma matricial de la siguiente manera:

$$\delta^L = \nabla_a C \odot \sigma'(z^L). \quad (\text{BP1a})$$

Otra ecuación necesaria es una que permita relacionar los errores entre diferentes capas.

Ya que realmente solo se conoce el error de la última capa, es necesario trasladar este

error a capas anteriores, para esto se utiliza:

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \quad (\text{BP2})$$

Donde w^{l+1} es la matriz de los *weights* para la capa $l + 1$. Si se multiplica δ^{l+1} -que ya se conoce gracias a la ecuación (BP1)- por la matriz w se está moviendo el error a una capa anterior, $\sigma'(z^l)$ es solo para darle una magnitud a este error.

Combinando la ecuación (BP1) y (BP2) se puede calcular el error para cualquier neurona en la red, ahora solo se necesita relacionar este error con el cambio en w y b .

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l. \quad (\text{BP3})$$

Esto significa que el error en una neurona es exactamente igual al cambio en el *bias* b de una neurona.

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l. \quad (\text{BP4})$$

La última ecuación establece que el cambio en un *weight* w es igual al vector de activación en la entrada de esta neurona multiplicada por el error.

Usando estas 4 ecuaciones se puede definir el algoritmo para entrenar una red de la siguiente manera:

1. Insertar un grupo de datos de entrenamiento.
2. Para cada ejemplo de entrenamiento meter los valores a la entrada de la primer

capa.

3. Para cada capa $l = 2, 3, \dots, L$ calcular $z^{x,l} = w^l a^{x,l-1} + b^l$ y $\sigma(z)$.
4. Calcular el error para la última capa con la ecuación (BP1).
5. Propagar hacia las capas anteriores este error usando (BP2).
6. Modificar los valores de w y a usando ***gradient descent*** y las ecuaciones (BP3) y (BP4)

3.2. Recurrent Neural Networks

El tipo de redes que se mostró al inicio de esta sección son llamadas *Feedforward Networks* por el modo que funcionan; se insertan datos y en un flujo en una sola dirección estos datos son convertidos en información. No tiene relevancia el orden en que se muestren los datos.

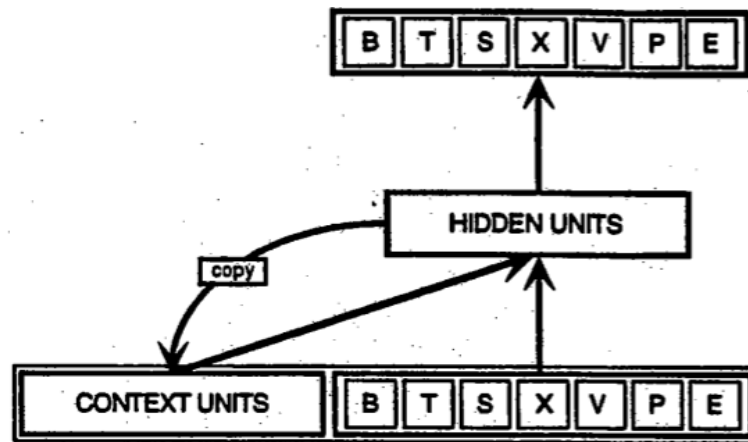


Figura 3.3: Estructura básica de una Red Neuronal Recurrente. Tomada de Servan-Schreiber et al. (1991)

En el caso que sea importante la secuencia en la que se insertan los datos existen las llamadas *Recurrent Neural Networks* (RNN). Estas guardan un estado que refleje el flujo histórico de los datos y lo utilizan como retroalimentación. A este estado se le va a llamar *hidden state* y será denotado con h_t donde t es un momento en el tiempo. La ecuación que define a este estado es la siguiente:

$$h_t = \phi(Wx_t + Uh_{t-1})$$

Donde h_t es el *hidden state* en el tiempo t , ϕ es una función que puede ser sigmoide o *tanh*, W es la matriz de *weights*, X_t es el vector de entrada en el tiempo t , U es la matriz de transición (lo mismo que *weights* pero para h) y h_{t-1} es el vector de hidden states anteriores.

Este tipo de redes utiliza una variación del algoritmo de *Backpropagation* para su entrenamiento conocido como *Backpropagation Through Time* (BPTT), la variación es que debe expandir la función de retroalimentación n veces antes de calcular y .

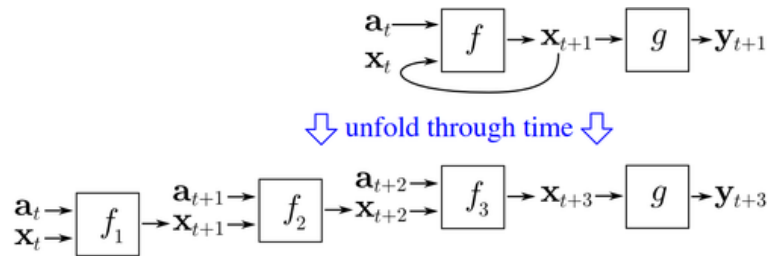


Figura 3.4: Algoritmo de BPTT. Tomada de Backpropagation through time - Wikipedia

El problema que surge con este tipo de redes es que es muy difícil conocer la importancia de un suceso muy remoto porque la información puede pasar por muchas operaciones y se puede ir reduciendo, haciendo muy lento el aprendizaje ya que los cambios son míni-

mos. Para resolver este problema surgen las llamadas *Long Short Term Memory Networks* (LSTM).

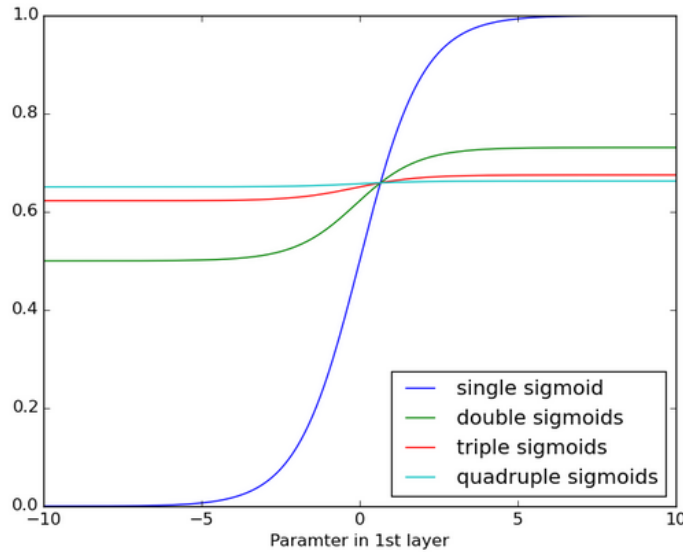


Figura 3.5: Efectos de aplicar una función sigmoide múltiples veces. Tomada de Recurrent neural network - Wikipedia

3.3. LSTM

Son un tipo de red que permite el aprendizaje incluso con más de 1000 pasos en el tiempo. Almacenan información dentro una celda, fuera del flujo normal de la red. En esta se puede escribir y leer información por medio de compuertas implementadas con la función sigmoide. Al igual que las neuronas, bloquean o permiten pasar cantidades de información dependiendo de sus propios *weights*, así las celdas aprenden cuándo permitir que la información sea escrita o leída. El siguiente diagrama representa el funcionamiento de una LSTM.

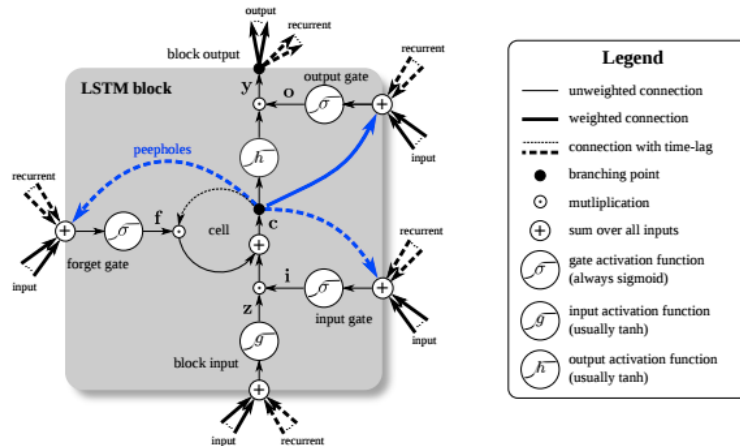


Figura 3.6: Representación de una LSTM. Tomada de Understanding LSTM Networks - Colah's blog

La información del *input* llega por debajo, se le aplica una función de aplastamiento como *tanh*. Posteriormente es multiplicada por la información que ya contenía la celda. Se actualiza la información de la celda en cada paso del tiempo, las compuertas pueden decidir que tanto olvidar, escribir o leer de la información.

3.4. Implementación de una LSTM para predicción de demanda

Una vez que se conocen las bases de una red neuronal, es fácil notar porque estas se pueden utilizar para predicción. La función de demanda a través del tiempo en el BSS de Ecobici resultaría imposible de determinar, pero puede aproximarse con una LSTM. A continuación se muestra la arquitectura de la red desarrollada usando la librería de Python **Keras**.

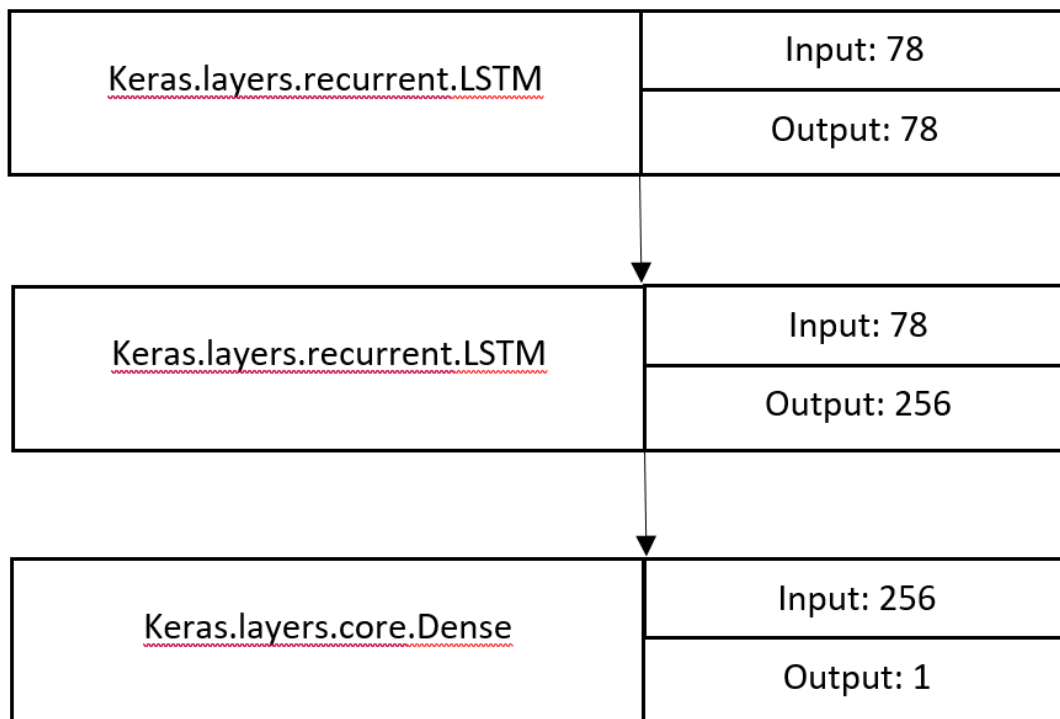


Figura 3.7: Representación del modelo de la red neuronal

A continuación se presentan las predicciones de la red para diferente número de *epochs*, en naranja es lo predicho por la red y azul las demandas reales. Cabe recalcar que no hubo una retroalimentación en la predicción para estas gráficas, esto quiere decir que lo predicho por la red no se utilizó para predecir el siguiente paso, en cambio se utilizaron solo los últimos 78 momentos reales.

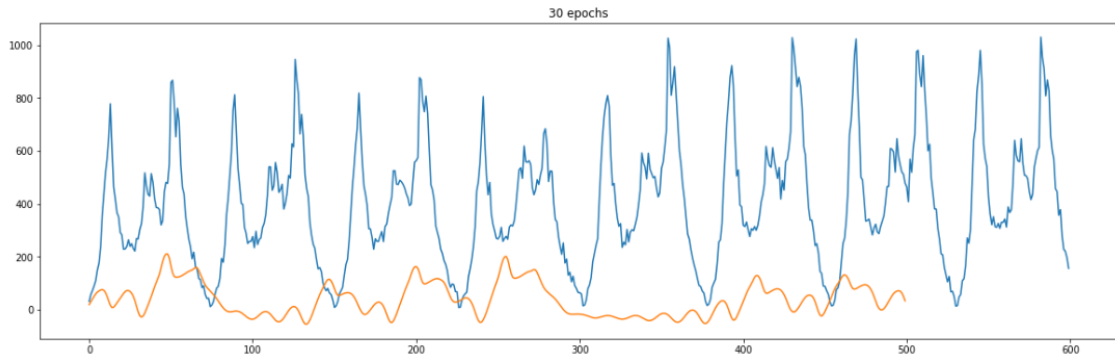


Figura 3.8: Red con 30 epochs de entrenamiento prediciendo demanda para todo el sistema

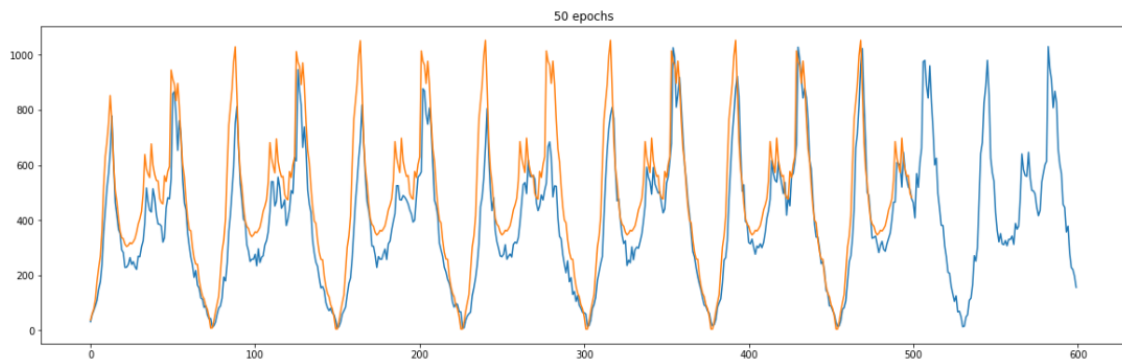


Figura 3.9: Red con 50 epochs de entrenamiento prediciendo demanda para todo el sistema

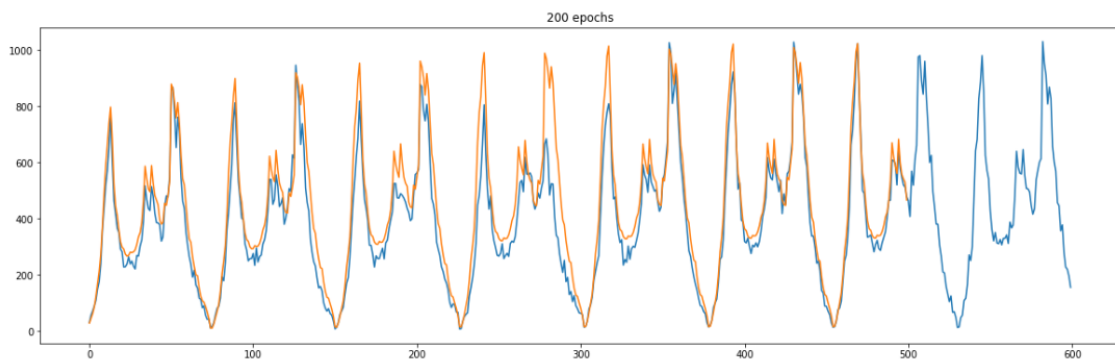


Figura 3.10: Red con 200 epochs de entrenamiento prediciendo demanda para todo el sistema

Si el mismo modelo es utilizado para predecir un día completo con retroalimentación de la red se obtiene el siguiente resultado. Esto significa que cada pronóstico que va

haciendo la red se vuelve a insertar en el vector *input* para predecir el siguiente paso

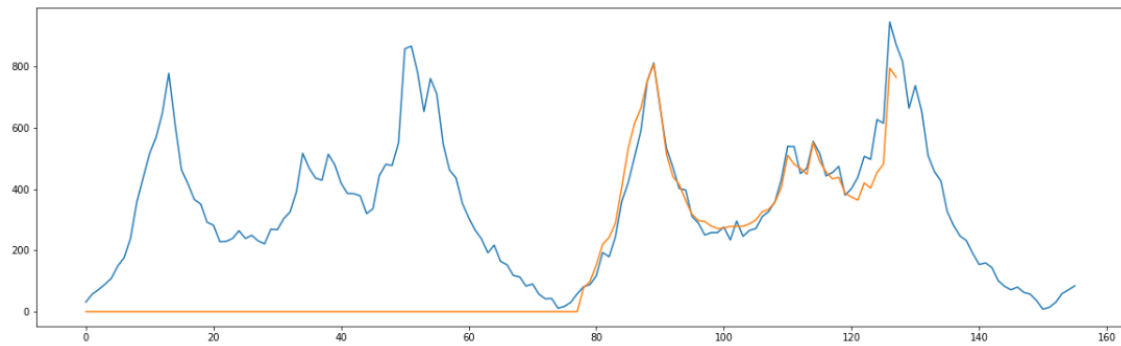


Figura 3.11: Predicción de casi un día con retroalimentación de la red

Estos son buenos resultados, en cambio si se quiere utilizar el mismo modelo para predecir la demanda en el cluster de Buenavista es notorio que difieren los resultados reales con los predichos.

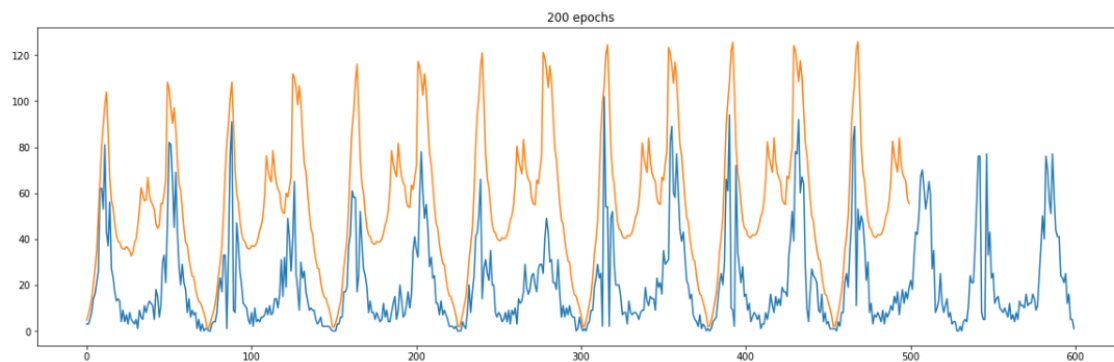


Figura 3.12: Predicción usando el modelo de todo el sistema para pronóstico de un cluster

4. Predicción de demanda para los clusters

En las secciones anteriores se muestran los clusters con los que se van a trabajar, así como una buena respuesta de la red neuronal para pronóstico de todo el sistema, por lo que esta sección busca juntar ambos supuestos y predecir la demanda en las próximas horas del clúster. A pesar de que se entrenaron 14 redes, una para cada clúster y epochs de 30, 100 y 200, no se mostrarán los resultados de todos para evitar saturar el documento.

Tomando como ejemplo el cluster 8 formado en las colonias Del Valle, Santa Cruz Atoyac y alrededores el modelo de 50 epochs de entrenamiento parece ser más adecuado si se comparan en base al **Error cuadrático medio** y **Error absoluto medio** entre los datos reales para los primeros 4 días hábiles de 2017 y los pronosticados.

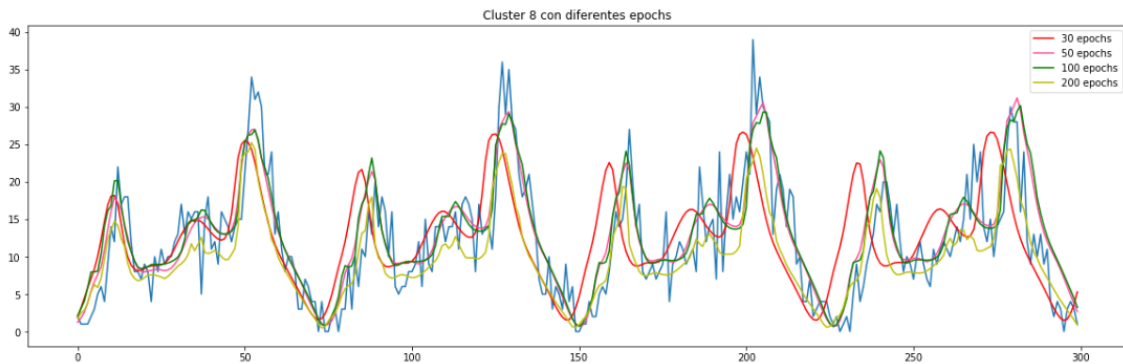


Figura 4.1: Redes neuronales del cluster 8 con diferentes epochs

	Métricas	
	Mean Square Error	Mean Absolute Error
30 epochs	47.86540178954771	5.39564008752505
50 epochs	16.52319792382705	3.1541734021902084
100 epochs	18.095638355582725	3.2299707897504173
200 epochs	25.22412443765576	3.80746980547905

Tabla 4.1: Métricas para diferentes epochs

La figura 28 muestra la predicción inmediata por ventanas de 78 datos, esto es que no hay una retroalimentación de lo predicho. Pero es claro que desde este momento parece que no es suficiente la información que arroja la red. En los picos que se forman a la hora de la salida laboral, los datos obtenidos están hasta un 20 % por debajo del real. Esto es grave especialmente si se considera que es en estos momentos donde resulta crucial tener datos confiables para soportar la carga en todo el sistema.

Si se quiere predecir en un momento dado las próximas horas se obtiene una función parecida a la real, pero nuevamente parece ser insuficiente para las demandas reales, patrón que se repitió para los demás clusters. Por ejemplo, tomando un día aleatorio del 2017 para el cluster 8 se pronostican las próximas 5 horas y se obtiene el siguiente resultado:

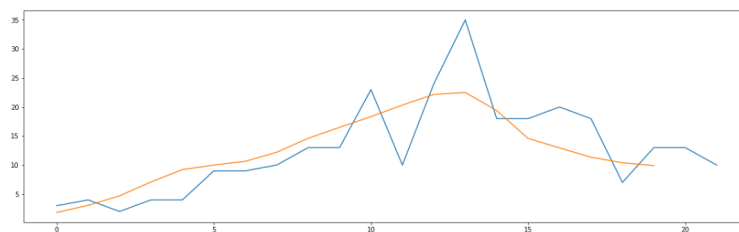


Figura 4.2: Predicción de próximas 5 horas para momento aleatorio del cluster 8

5. Conclusión y posibles mejoras

La red neuronal es capaz de generar un pronóstico de la demanda cercano al real, pero es insuficiente si se considera que la diferencia puede resultar en decenas de bicicletas o espacios desacertados.

Se pueden realizar muchos cambios que mejoren la precisión de la red, algunos pueden ser:

- Añadir como dimensión en la red datos del servicio meteorológico. Como menciona Gebhart (2017) las condiciones climatológicas influyen directamente en la demanda de este tipo de sistemas
- Agrupamiento dinámico. Algunas estaciones parecen comportarse diferente en diversas horas del día.
- Otros métodos de agrupamiento o restricciones de distancia. Hay casos en los que estaciones quedan agrupadas en zonas que parecen lejanas en la realidad.
- Explorar otras arquitecturas para la red neuronal. Se pueden añadir más capas para que aprenda representaciones temporales de mayor nivel.

Considero que no se ha terminado de explorar la opción de pronosticar usando redes neuronales, con algunas mejoras podrían obtenerse resultados más acertados y útiles para los BSS.

Referencias

- [1] Ahmadreza F., Hampshire R., Lavanya M. y Naveen E. (2017) *An empirical analysis of bike sharing usage and rebalancing: Evidence from Barcelona and Seville*, Transportation Research Part A, Elsevier.
- [2] Brendan J. y Delbert D. (2007) *Clustering by passing messages between data points*, Science.
- [3] Estadísticas de Ecobici *Traffic Prediction in a Bike-Sharing System*, Shanghai Jiao Tong University, Shanghai, China.
- [4] Gebhart K. y Noland R. (2014) *The impact of weather conditions on bikeshare trips in Washington, DC*, Springer Science+Business Media, Nueva York, Estados Unidos.
- [5] Hazel G. y Miller D. (2007) *Desafíos de las Megaciudades*, [http://www.aan.siemens.com/chile/e-brochures/Documents/Desafiosde las Megaciudades - Siemens.pdf](http://www.aan.siemens.com/chile/e-brochures/Documents/Desafiosde%20las%20Megaciudades%20-%20Siemens.pdf).
- [6] Kasambara A. (2017) *Practical Guide To Cluster Analysis in R*, STHDA.
- [7] Li Y., Zheng Y., Zhang H. y Chen L. (2014) *Traffic Prediction in a Bike-Sharing System*, Shanghai Jiao Tong University, Shanghai, China.
- [8] Nielsen M. (2015) *Neural Networks and Deep Learning*, Determination Press.
- [9] Paul DeMaio. (consultado en *The bike sharing world map* Julio de 2017 <https://maps.google.com/maps/ms?ie=UTF8&hl=en&om=1&msa=0&msid=10422>

7318304000014160.00043d80f9456b3416ced&ll=43.580391,-
42.890625&spn=143.80149,154.6875&z=1&source=embed&dg=feature

- [10] Raviv T. y Kolka O. (2013) *Optimal inventory management of a bike-sharing station*, Taylor & Francis Ltd.
- [11] Schuijbroek J., Hampshire R. y Hoeve W. (2013) *Inventory Rebalancing and Vehicle Routing in Bike Sharing Systems*, Carnegie Mellon University, Estados Unidos.
- [12] Servan-Schreiber, D., Cleeremans, A., y McClelland, J. L. (1991) *Graded state machines: The representation of temporal contingencies in simple recurrent networks.*, Machine Learning.
- [13] Tan, P.-N., Steinbach, M. y Kumar, V. (2006) *Introduction to Data Mining*, Pearson Addison-Wesley.
- [14] Vogel P., Greiser T., Mattfeld. (2011) *Understanding Bike-Sharing Systems using Data Mining: Exploring Activity Patterns*, Elsevier.
- [15] Zhao Z., Chen W., Wu X., Chen P. y Liu J. (2017) *LSTM network: a deep learning approach for short-term traffic forecast*, The Institution of Engineering and Technology.