INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

APRENDIZAJE DE MÁQUINA

AUGUSTO HERNANDEZ GALLEGOS

112835

RECONOCIMIENTO DE IMÁGENES PARA LA MEJORA DE LA OPERACIÓN DE UNA EMPRESA DE CONSUMO

Motivación del trabajo

En México hay más de 1.5 millones de tiendas tradicionales (tenditas) que ponen a la venta gran cantidad de artículos para consumo, desde botanas y galletas hasta perecederos como jamón y queso. El inventario para venta con el que cuentan estas tiendas es provisto principalmente por medio de la entrega directa y por la compra en las centrales de abasto. El alto costo operativo y las inversiones necesarias para la construcción de un sistema de distribución directa deriva en que sean pocas las empresas que cuenten con este canal de venta. Asimismo, las empresas que cuentan con estos sistemas de distribución tienen miles de vendedores en las calles con tecnología que pueden digitalizar lo que observan en sus visitas diarias y esta información a su vez pueda transformarse en estrategias que mejoren las ventas o hagan más eficiente la operación.

Para este trabajo, nos enfocamos en una situación en particular. Dentro de estas tienditas, hay estantes o refrigeradores donde se acomodan los productos dependiendo de su categoría y fabricante. En el caso de las categorías más grandes como refresco, botanas y galletas, los fabricantes proveen al tendero de los estantes para colocar la mercancía (que tienen el nombre de racks). Dependiendo del nivel de venta del cliente estos racks pueden ser más chicos o más grandes y cada tipo de rack tiene un acomodo predefinido, llamado planograma. El diseño de este se hace con base en la rotación estimada de los productos. Bajo esa lógica, los productos que tienen una mayor rotación tienen más espacios en el rack, llamados frentes. El planograma de cada tipo de rack determina un valor estimado de cada uno considerando el número de frentes de cada producto, la rotación estimada y el valor de cada unidad.

Imagen 1. Rack en una tienda tradicional

Ahora bien, usando reconocimiento de imágenes es posible detectar qué productos están presentes en el rack y en caso de que fuera posible contar cuántos frentes reconoce por producto podría permitir hacer un inventario del rack con una foto. Dicho de otra manera, cuando el vendedor llega a surtir la tienda puede tomar una foto de cómo está el rack y eso le permite ver qué productos tienen más frentes faltantes y hacer un aproximado de qué productos deberían tener más inventario. Asimismo, una vez que terminara su visita en la tienda podría tomar una nueva foto que podría comparar con la foto inicial de la primera visita para entender el desplazamiento de los productos y el nivel de inventarios (variable sumamente compleja en la industria de consumo). De manera complementaria, esta clasificación a través de imágenes permitiría entender aún mejor el comportamiento de los clientes y tener variables mucho más sencillas de manejar una vez procesadas las imágenes. Un primer paso para lograr el objetivo presentado es usar el reconocimiento de imágenes para reconocer de qué producto es la imagen. Para ello, usaremos tres métodos de clasificación supervisada: una red neuronal simple, una máquina de soporte vectorial y el método de k nearest neighbor.

Metodología

Construcción de las imágenes

Para la realización del trabajo se tomaron cerca de dos mil fotos de cuatro distintas marcas de Gamesa: Emperador, Cremax, Marias y Chokis. Estas fotos se tomaron en distintas tiendas de canal tradicional, conveniencia y supermercados.

En el levantamiento de fotos se hicieron dos observaciones importantes. Primero, los productos con una mayor distribución, es decir, con una mayor presencia en las tiendas tuvieron una mayor cantidad de imágenes capturadas. Segundo, para tomar las imágenes para poder entrenar la red neuronal era necesario ser cuidadoso de no mezclar diferentes marcas en una foto, de otro modo la red no iba a poder aprender correctamente. Sin embargo, en la operación cotidiana los productos no están del todo separados. Esto por consiguiente implicaría que para poder hacer un ejercicio efectivo de reconocimiento de imágenes los vendedores tuvieran que ser demasiado cuidadosos en la ejecución de la tarea o que el reconocimiento de imágenes se complementara con una técnica, en un paso adicional, que pudiera separar en una imagen distintos productos a reconocer. Esta última tarea, podría ser elaborada mediante el uso de máquinas de soporte vectorial que muestran ser más efectivas para clasificar conjuntos no separables linealmente como serían imágenes con distintas formas y acomodos. El objetivo del trabajo no cubre analizar esta situación, aunque vale la pena mencionarla.

Las imágenes originales obtenidas tienen una resolución de 4128x2322 pixeles. Asimismo, para procesar las imágenes, se generan vectores con los pixeles y esos vectores son los que son procesados por la red neuronal. Por ello, conservar las imágenes con un pixelaje tan alto aumenta los costos computacionales al mismo tiempo que aumenta la complejidad del modelo. Considerando esto, el pixelaje de las imágenes se redujo a 80x45 pixeles usando la librería “magick” de R. Con esta librería y la función de resize se ajustaron las dos mil imágenes del modelo. Es de considerarse también si hay un número óptimo de pixeles a los que hay que ajustar las imágenes para que la precisión y complejidad del modelo se encuentren en niveles óptimos. Probablemente una imagen de alta resolución arroje los mismos resultados que una imagen de baja resolución, por varias razones. Por ejemplo, los empaques de los productos tienen una superficie

homogénea que hace que gran parte de la imagen tenga la misma secuencia de pixeles. Por lo tanto, en ese tipo de situaciones la alta y baja resolución probablemente contengan la misma información.

Imagen 2. Foto original

```
In [10]: plt.imshow(imagen_prueba)
Out[10]: <matplotlib.image.AxesImage at 0x7fa6762a1f10>
```

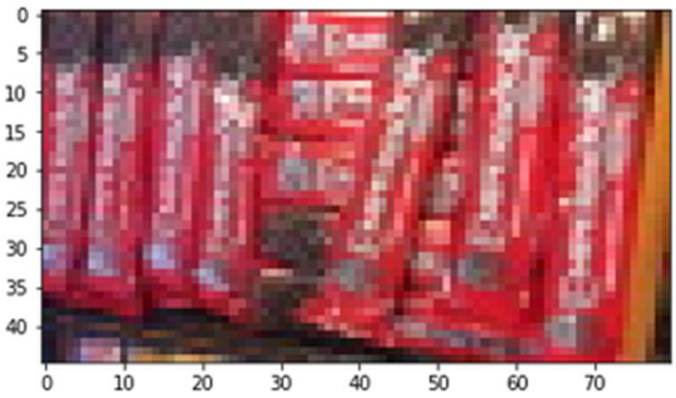

Imagen 3. Foto con pixelaje reducido

Posteriormente, las imágenes se ajustaron usando numpy en python. Las imágenes son leídas por la librería Image y se generan en un arrays de la forma (45, 80, 3). Usando el comando reshape convertimos la imagen a un array de la forma (1, 10800) y todas las imágenes se agrupan en una solo array que termina siendo de (2000, 10800) donde las filas son el número de fotos y las columnas son

las dimensiones del array de cada imagen. El rango de los valores del array es de 0 a 255, ya que los datos son de tipo uint8 en esta conversión de numpy. Este primer array, es el array de imágenes. Por otro lado, para el caso de la red neuronal, se crea un array de etiquetas con 4 columnas, una para cada categoría y en la que cada entrada tiene asignado un 1 en la categoría correspondiente y 0 en el resto. Para la máquina de soporte vectorial y el K-Nearest neighbor se creó un vector de etiquetas en el que se número del 1 al 4 dependiendo del producto.

De manera complementaria, la recolección de imágenes se programó de tal manera que permite que se ingresen nuevas imágenes, se conviertan y se entrenen los modelos.

Construcción de los modelos: Red Neuronal

Para poder analizar las imágenes construimos una red neuronal con dos capas escondidas y tres variantes en las neuronas de cada capa: la primera, con 500 neuronas en cada capa; la segunda, con 256 neuronas en cada y, la última, de 50 neuronas en cada capa. El número de inputs de la red es la dimensión de las columnas de la matriz de imágenes y el output de la red es el número de categorías que en este caso son las cuatro marcas de las que se reconocen las imágenes.

Para correr la red se utilizó la librería tensorflow en la que se seleccionó un parámetro de aprendizaje de 0.1. Asimismo, también se seleccionó una función de pérdida softmax_cross_entropy. La función softmax es una generalización del modelo logit hacia diferentes clases y la entropía cruzada es nuestra funciónde costo que penaliza el error. Asimismo, se entrenó la red con 10 y 50 iteraciones.

Construcción de los modelos: Máquina de soporte vectorial

De igual forma, otra herramienta que puede resultar útil para la clasificación de imágenes es una máquina de soporte vectorial. Sin embargo, a diferencia de la red neuronal, el vector de etiquetas clasifica las fotos del 1 al 4. Para que el

modelo funcione, cambiamos el kernel original, por un kernel RBF que permite ajustar mejor clasificaciones no lineales.

Construcción de los modelos: K nearest neighbor

Por último, usamos un modelo de K nearest neighbor para clasificar las imágenes con un vector de etiquetas similar al de la máquina de soporte vectorial. Asimismo, usamos 4 clusters en el modelo considerando que estabsmos buscando la clasificación de 4 tipos de imágenes

Implementación de los modelos

Se dividieron las imágenes en un conjunto de entrenamiento y de validación de 75% y 25% respetando la proporción de las imágenes por marca. Una vez separados ambos conjuntos se normalizaron y transformaron usando el paquete preprocessing de Python. Esto para reducir el espacio de los arrays de las imágenes que iban de 0 a 255 y dejarlos entre 0 y 1 con media cero y varianza unitaria.

Una vez procesados los datos, se entrenaron los tres modelos y se calculó su accuracy (proporción de predicciones acertadas sobre el total de predicciones) y este resultado se usó para comparar los modelos.

Resultados

Red Neuronal

Primero se corrió el una red neuronal con tres distintos número de neuronas (20, 256 y 500 neuronas) usando 10 iteraciones. El accuracy del modelo se observó de la siguiente forma: el modelo con 20 neuronas tuvo un accuracy de 0.692 en el conjunto de entrenamiento y 0.439 en el de validación. Por otro lado, el modelo con 256 neuronas tuvo un accuracy de 0.814 y 0.519 en el conjunto de entrenamiento y validación respectivamente. Por último, en el modelo con 500 neuronas se obtuvo un accuracy de 0.771 y 0.492 en el conjunto de entrenamiento y validación respectivamente.

Posteriormente, se corrieron los mismos modelos, pero usando 50 iteraciones, ya que, por las dimensiones de la foto como matriz (10800), la computadora no podía procesar un mayor número de iteraciones. Los resultados de estos modelos fueron los siguientes: el modelo con 20 neuronas tuvo un accuracy de 0.995 en el conjunto de entrenamiento y 0.510 en el de validación. Por otro lado, el modelo con 256 neuronas tuvo un accuracy de 0.997 y 0.5284 en el conjunto de entrenamiento y validación respectivamente. Por último, en el modelo con 500 neuronas se obtuvo un accuracy de 0.999 y 0.510 en el conjunto de entrenamiento y validación respectivamente.

Máquina de Soporte Vectorial

En el caso de la máquina de soporte vectorial se obtuvo un accuracy de 0.997 en el conjunto de entrenamiento y de 0.567 en el conjunto de validación

K Nearest Neighbors

En el caso del K Nearest Neighbors se obtuvo un accuracy de 0.957 en los datos de entrenamiento y de 0.435 en el conjunto de validación.

Conclusiones

La implementación de técnicas de reconocimiento de imágenes pueden permitir una mejora en la ejecución de los vendedores, integrarse como variable para la predicción de ventas y en el diseño de los medios de exhibición que permitan maximizar la venta de los productos en un portafolio de productos.

En los modelos utilizados se observó un accuracy de muy alto en el conjunto de entrenamiento comparado con el conjunto de validación, con esto es claro que es necesario un mayor número de imágenes que entrenen correctamente el modelo y puedan predecir mejor el conjunto de validación.

En las redes neuronales, al aumentar el número de iteraciones se observó una mejora significativa en el accuracy de la predicción de los datos de entrenamiento al pasar de 0.692, 0.814 y 0.771, en cada uno de los tres modelos mencionados, a

0.99 en todos los casos. Asimismo, el número de neuronas también derivó en una mejora del accuracy cuando paso de 20 a 256, pero una disminución cuando pasó de 256 a 500. Posiblemente por un aumento en la complejidad del modelo.

Por otro lado, en el caso de la máquina de soporte vectorial y el K Nearest Neighbor se observó la misma diferencia en accuracy entre la predicción con el conjunto de validación y el de entrenamiento. Asimismo, se observó que el K Nearest Neighbor tuvo el menor accuracy de todos los modelos en el conjunto de entrenamiento con un 0.435.

Para poder entrenar mejor el modelo es necesario usar un mayor número de imágenes. De otro modo, es muy probable que el modelo se sobreajuste por la falta de datos y esto genere una alta varianza. Asimismo, es necesario reducir las dimensiones de los pixeles que tiene cada imagen para reducir la complejidad del modelo y que de igual forma no se sobreajuste.

Por otro lado, los productos con mayor distribución tengan una mayor cantidad de imágenes para entrenar y llegue a una buena precisión mientras que los productos con una menor distribución sean los que tengan una mayor cantidad de casos de mala clasificación por la menor cantidad de ejemplos con los que pueden entrenar los modelos comparado con los productos más distribuidos

Asimismo, agregar técnicas complementarias puede permitir un siguiente paso para poder reconocer más de un producto dentro de una imagen. Por ejemplo, complementándolo con máquinas de soporte vectorial. En la figura 4 se observa una propuesta de cómo podría segmentarse una imagen en distintas etapas para poder hacer varias clasificaciones en una sola imagen. En este ejemplo el Stage 1 representa una imagen completa que puede ser partida en 3 por un primer algoritmo de clasificación y con lo que se llega al Stage 2. Por último, en el Stage 3 cada imagen se puede dividir entre productos y una vez hecha toda la segmentación hacer la clasificación de todos los subconjuntos e incluso hacer conteos de imágenes similares. En este caso el alta de resolución de las imágenes podría ser ya un factor de suma importancia, ya que permitiría hacer más con una

sola imagen y la segmentación en subgrupos de imágenes reduciría la complejidad de los modelos al mismo tiempo que aprovecha todos los pixeles disponibles.

Figura 4. Una propuesta hacia el futuro

Bibliografía

- https://github.com/aymericdamien/TensorFlow-Examples/blob/master/notebooks/3_NeuralNetworks/neural_network_raw.ipynb
- http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html
- http://scikit-image.org/docs/dev/user_guide/numpy_images.html
- http://cs231n.github.io/linear-classify/