Redes Neuronales Profundas: Trabajo Práctico 1

Respuestas

El código realizado para la resolución de los ejercicios está disponible en github (branch *lpineda*)

Ejercicio 1

Para leer el dataset completo se genera recursivamente una lista de *strings* con las rutas de las imágenes. Luego, estas son leídas utilizando la función *reshape_image*, la cual descarta una fila o una columna si el alto o el ancho tienen dimensiones impares, y centra la imagen en un rectángulo negro de las medidas dadas. Dado que el dataset completo no cabe en memoria, solo se lee un subdirectorio elegido al azar para realizar las pruebas.

Ejercicio 2

En este ejercicio se toman imágenes directamente del dataset, sin el procesamiento realizado en el ejercicio 1. Dado que se deben extraer parches en posiciones aleatorias de la imagen, la subimagen podría ser seleccionada de uno de los bordes negros, lo cual no agregaría información relevante. Como el tamaño de los parches es muy pequeño, no se incluye ninguna captura. La función está implementada con el nombre *extract_window*.

Por otro lado, para normalizar la imagen con $\mu = 0$ y $\sigma^2 = 1$, a cada pixel de la imagen se le sustrae la media y se lo divide por la desviación estandar. Esto es, para una imagen I:

$$\forall p_{i,j} \in I, \qquad \hat{p}_{i,j} = \frac{(p_{i,j} - \mu_I)}{\sigma_I^2}$$

El resultado de este procesamiento se puede observar en la figura 2. La imagen transformada es inteligible puesto que al normalizar cada canal, los valores enteros de los pixeles en el rango $[0, 255]$ son transformados a valores flotantes, y al tener una distribución de probabilidad con $\mu = 0$ es

de esperar la existencia de valores negativos. Estos son casteados a enteros por la librería utilizada[1].

Figura 1: Imagen original.

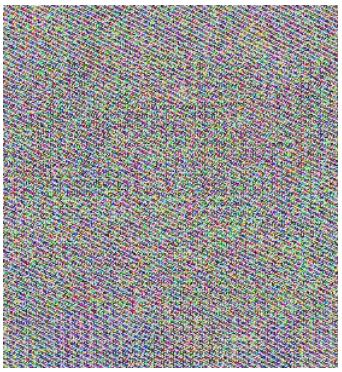

Figura 2: $\mu = 0 \quad \sigma^2 = 1$

Sin embargo, el hecho que la imagen sea inteligible no quiere decir que la información se haya perdido. Esto se puede comprobar al transformar los valores de los pixeles al rango $[0, 255]$ mediante la función:

$$f(x) = int\{255\frac{x-min(I)}{max(I)-min(I)}\}$$

donde nuevamente I es una imagen dada. Esta función está implementada bajo el nombre *quantize*, y el resultado de su aplicación a los pixeles de la imagen 2 (en su versión de valores flotantes) puede observarse en la figura 3.

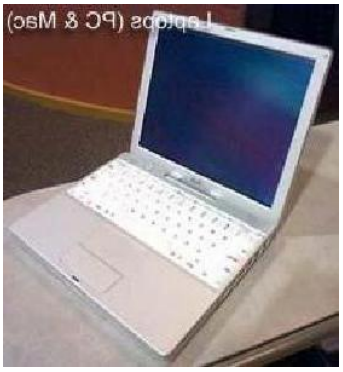

Figura 3: Imagen cuantizada

[1]Python Imaging Library (PIL): http://www.pythonware.com/products/pil/

Si bien a primera vista las imágenes originales y transformadas pueden parecer iguales, en base al histograma podemos observar que la transformación aplicada hace que la distribución de los pixeles varíe en cada canal, siendo la segunda más uniforme.

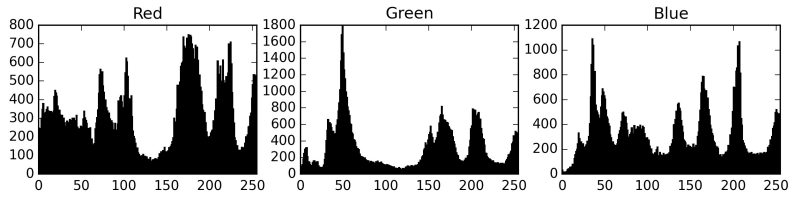

Figura 4: Histograma de la imagen original

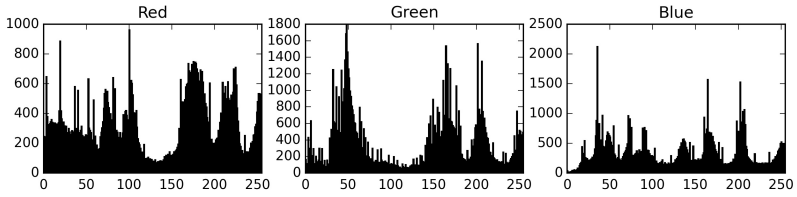

Figura 5: Histograma de la imagen cuantizada

Ejercicio 3

Las imagenes naturales (las imagenes que los humanos estamos acostumbrados a ver, como paisajes, animales, etc) contienen información redundante dada la alta correlación entre pixeles adjacentes. Si utilizamos un dataset de imágenes para entrenar algún modelo, lo ideal sería que las entradas contengan la menor cantidad de información repetida posible; el objetivo de aplicar *whitening* sobre cada imagen de un dataset es hacer que los datos con los que entrenamos la red neuronal no tengan información redundante.

Dados m vectores de características $\mathbf{x}$ (nuestro dataset) con media cero $\mu_{\mathbf{x}_i} = 0$, la matriz de covarianza está dada por:

$$\mathbf{\Sigma} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}\mathbf{x}^T$$

Si queremos que los datos no estén correlacionados, debemos transformar el dataset de forma que su matriz de covarianza $\mathbf{\Sigma}$ sea una matriz diagonal. Esta transformación se puede expresar lograrse pre-multiplicando cada vector de características por la matriz $\mathbf{\Phi}^T$, donde las columnas de $\mathbf{\Phi}$ son los

eigenvectores de la matriz de covarianza $\mathbf{\Sigma}$. Podemos expresar la matriz de covarianza diagonalizada como:

$$\mathbf{\Phi}^T \mathbf{\Sigma} \mathbf{\Phi} = \mathbf{\Lambda}$$

donde $\mathbf{\Lambda}$ es una matriz de ceros con los eigenvalores de la matriz de covarianza en su diagonal. Luego, la matriz de covarianza de $\mathbf{y} = \mathbf{\Phi}^T \mathbf{x}$ es la matriz diagonal $\mathbf{\Lambda}$.

Aplicar *whitening* a un dato consiste en hacer que su matriz de covarianza tenga la forma $\mathbf{\Lambda} = \alpha \mathbf{I}$. Usando el hecho que:

$$\mathbf{\Lambda}\mathbf{\Lambda}^{-1} = 1 \rightarrow \mathbf{\Lambda}^{-1} = \mathbf{\Lambda}^{-1/2}\mathbf{I}\mathbf{\Lambda}^{-1/2} \rightarrow \mathbf{I} = \mathbf{\Lambda}^{-1/2}\mathbf{\Lambda}\mathbf{\Lambda}^{-1/2}$$

y reemplazando $\mathbf{\Lambda}$ tenemos que:

$$\mathbf{\Lambda}^{-1/2}\mathbf{\Phi}^T\mathbf{\Sigma}\mathbf{\Phi}\mathbf{\Lambda}^{-1/2} = \mathbf{I}$$

Finalmente, la transformación $\mathbf{z}$ para *blanquear* los datos esta dada por:

$$\mathbf{z} = \mathbf{\Lambda}^{-1/2}\mathbf{\Phi}^T\mathbf{\Sigma}\mathbf{x}$$

Esto se puede comprobar al calcular la matriz de covarianza de $\mathbf{z}$

$$\mathbf{\Sigma} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{z}\mathbf{z}^T = \frac{1}{m}\sum_{i=1}^{m} \mathbf{\Lambda}^{-1/2}\mathbf{\Phi}^T\mathbf{x}\mathbf{x}^T\mathbf{\Phi}\mathbf{\Lambda}^{-1/2} = \mathbf{I}$$

Ejercicio 4

Aplicar un procedimiento similar al del ejercicio 2 en las imágenes del dataset de palabras manuscritas produce resultados similares. Aunque el cambio es muy sutil, podemos observar que la imagen normalizada se aclara.

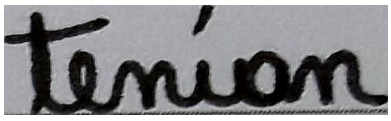

Figura 6: Imagen original.

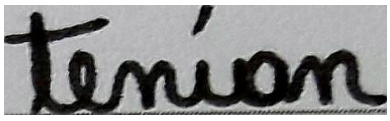

Figura 7: $\mu = 0 \quad \sigma^2 = 1$

Al analizar los histogramas, vemos que los picos con la mayor concentración de pixeles se desplazan a la derecha, lo que resulta en un aumento general de brillo de la imagen.

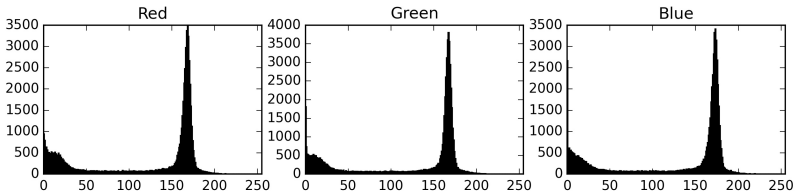

Figura 8: Histograma del texto manuscrito original

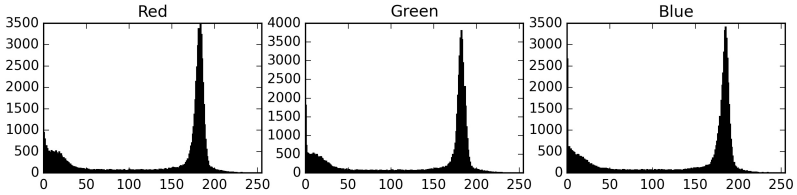

Figura 9: Histograma del texto manuscrito normalizado y cuantizado

Ejercicio 6

El texto es separado en oraciones, y cada oración es transformada a un vector de números. Dado el vector de ocurrencia de las palabras del texto completo ordenado por frecuencia, la transformación mapea a cada oración un vector donde se cuentan las apariciones de cada palabra más frecuente en dicha oración. Luego, se reduce la dimensionalidad de los datos utilizando PCA.

Se pueden observa puntos distantes en ambas distribuciones. Se separaran las sentencias correspondientes a estos puntos en los archivos de texto *rare_top.txt* y *rare_low.txt*.

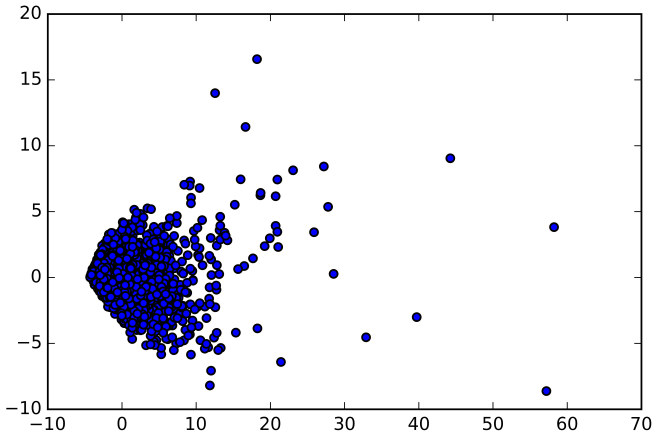

Figura 10: Scatter PCA vector de mayores frecuencias.

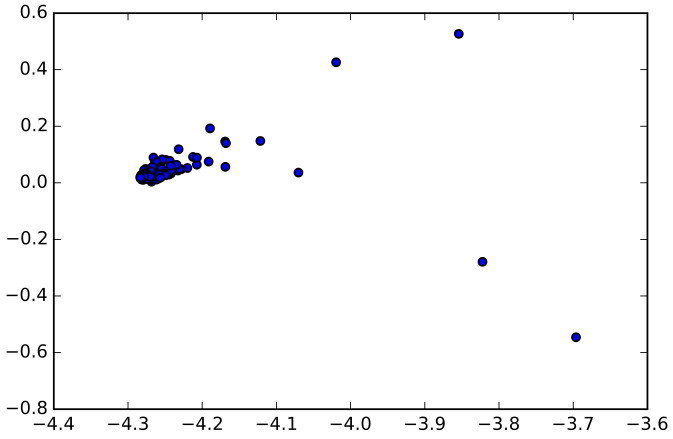

Figura 11: Scatter PCA vector de inversas de menores frecuencias.