PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
IIC-2613

TAREA 4: REDES NEURONALES

LA TAREA PUEDE SER REALIZADA EN PAREJAS

Fecha de Entrega: 10:00AM hrs, Miércoles 2/12 (Siding).

1 Objetivo

Muchach@s, en esta tarea tendrán la oportunidad de experimentar con redes neuronales tipo feedforward con capa oculta. En particular, usarán este tipo de técnica para clasificar imágenes. Para esto usarán un sub-conjunto del set de datos MIT-67. Este set contiene imágenes de 67 tipos de escenas, sin embargo, para reducir los tiempos de procesos, en esta tarea experimentarán con un subconjunto de 6 de estas escenas: restaurant, auditorium, classroom, closet, movietheater y bar.

2 Atributos

Como comentamos en clase, redes neuronales de tipo profundo (deep learning) han surgido recientemente como una poderosa herramienta para aprender en forma automática atributos (features) que permiten aumentar significativamente el rendimiento de diversas tareas de clasificación, tales como el reconocimiento de voz, texto, o imágenes. Como también discutimos en clases, en general, el entrenamiento de estas redes requiere ajustar millones de parámetros, lo cual es altamente demandante y requiere recursos computacionales que escapan los disponibles en el curso. Afortunadamente, gracias a los contactos de su profesor con el cluster del grupo de investigación en Inteligencia de Máquina (`grima.ing.puc.cl`), cada una de las imágenes usada en esta tarea fue previamente procesada por una red neuronal profunda para obtener un conjunto de atributos para la clasificación.

Apuesto a que pueden adivinar la red seleccionada,...., exacto!... se trata de Alex's Net, la arquitectura vista en clases. Esta red consiste de 8 capas, pero para efectos de esta tarea se eliminan las 2 últimas, quedando como salida los atributos provistos por la capa 6, o FC6 (**F**ully **C**onnected layer 6). La salida de esta capa consiste de 4096 neuronas, por tanto, para cada imagen de entrada se obtiene un vector de atributos de 4096 dimensiones.

3 Preprocesamiento: Ventana deslizante y max-pooling

Una de las características relevantes del mundo visual es la alta coherencia local de los patrones visuales. Por ejemplo, al considerar imágenes de bares, existen estructuras espacialmente localizadas que se repiten, como pisos, mesones, áreas con botellas, etc. Para poder capturar esta información local se utiliza un estrategia denominada ventana deslizante (sliding window), que consiste en deslizar sobre la imagen una ventana espacial según un paso regular en la dirección horizontal y vertical. Así es posible obtener vectores de atributos de distintas áreas de cada imagen que denominaremos patches, tal como muestra la Figura 1.

Dado que las imágenes usadas en esta tarea tienen una resolución de 256x320 pixeles, al utilizar una ventana de 64x64 pixeles y un paso de 32 pixeles, se obtiene en cada imagen un total de 7x9=63 patches. Cada uno de estos patches se ingresa a Alex's net para obtener un vector de 4096 dimensiones por cada patch. Luego, para obtener el vector de atributos de la imagen completa, se integran los vectores de cada patch usando una estrategia denominada max-pooling. Ésta consiste en tomar la máxima respuesta para cada componente del vector de atributos. Por ejemplo, si en lugar de 4096 cada vector fuera de 5 dimensiones y tuvieramos 3 patches, el resultado de max-pooling para los siguientes 3 vectores sería: maxpool {(**10**, 3, 4, 5, 6); (3, **21**, 4, 5, 6); (1, 3, **80**, **100**, **50**)}=(10, 21, 80, 100, 50). En el caso de la tarea, para cada imagen el proceso de max-pooling entrega un vector de 4096 dimensiones, los cuales están disponibles en

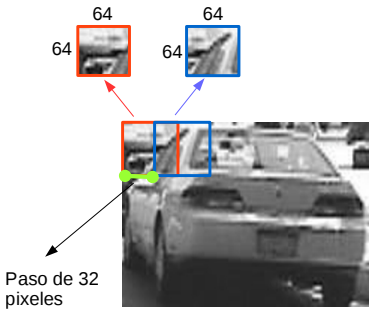

Figure 1: Ventana deslizante usando patches de 64x64 pixeles y un paso de 32 pixeles.

el sitio web del curso en los archivos: FC6Train.zip y FC6Test.zip para entrenamiento y test, respectivamente.

3.1 Entrenamiento del Clasificador

Estamos ahora listos para la acción. Los software recomendados para esta tarea son: Torch (http://torch.ch/) y Keras: http://keras.io/. Personalmente, tengo instalado Torch en mi computador personal y funciona estupendo. El principal inconveniente es que Torch usa como lenguaje de programación Lua, con el cual pueden estar menos familiarizados. Por su parte, Kera usa Python. En cualquier caso, para los fines de esta tarea, el lenguaje de programación no es gran problema pues principalmente necesitan usar funciones de librerías. En el siguiente link podrán encontrar un tutorial de como ejecutar una red neuronal con 1 capa oculta (multilayer perceptron) usando Torch: github.com/torch/tutorials/blob/master/2_supervised/2_model.lua. Aca para Keras: keras.io/examples/.

Adicionalmente, existe una gran cantidad de software que implementa redes neuronales, cualquier implementación que les acomode esta ok, Torch y Keras son sólo una recomendación. El software Weka (www.cs.waikato.ac.nz/ml/weka/) también puede ser una alternativa posible, la ventaja es que tiene una interface de usuario muy amigable en base a menús, pero puede ser algo lento. El ayudante del curso recomienda el toolbox de python de Issam Laradji disponible en: github.com/IssamLaradji/NeuralNetworks.

Usando la plataforma de software de su elección y el set FC6Train.zip, entrene una red neuronal con 1 capa oculta, según lo siguiente:

- Para la capa de entrada utilice tantas neuronas como atributos, i.e., 4096.
- Para la capa de salida pruebe con 2 posibles codificaciones de su elección.
- Para la capa oculta ajuste empíricamente el número de neuronas.
- Entrene las redes utilizando gradientes estocástico, i.e., el método incremental visto en clases.

3.2 Prueba del Clasificador

Tal como en la tarea anterior, reporten sus resultados de clasificación utilizando una matriz de confusión (http://en.wikipedia.org/wiki/Confusion_matrix). En su reporte conteste las siguientes consultas.

- ¿Cómo determinó el número de neuronas de la capa oculta?
- ¿Qué criterio utilizó para terminar de iterar el proceso de entrenamiento?.
- ¿Cuál es la exactitud promedio en cada set?. Observa sobreajuste, comente.
- ¿Qué categoría obtiene el mejor rendimiento?, ¿Cuál el más bajo?. Indique razones que puedan justificar cada resultado. Para esto mire las imágenes reales, las cuales están en los archivos ImageTrain.zip e ImageTest.zip.
- Compare el resultado obtenido para las 2 codificaciones seleccionadas para la capa de salida.