PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DIPLOMADO BIG DATA

ACTIVIDAD PRÁCTICA 1: ALGORITMOS DE APRENDIZAJE DE MÁQUINA

Material de preparación evaluación y laboratorio 1.

1 Objetivo

Estimad@s, en esta actividad tendrán la oportunidad de poner en práctica los conocimientos que han aprendido en clases. En particular, probarán el rendimiento de diferentes clasificadores para reconocer imágenes de distintos tipos de escenas. Para esto usarán el set de datos MIT-67, el cual contiene imágenes de 67 tipos de escenas.

2 Información General

2.1 Espacio de Características (Feature space)

Como comentamos en clase, redes neuronales de aprendizaje profundo (deep learning) han surgido recientemente como una poderosa herramienta que permite aprender características (features) que permiten aumentar significativamente el rendimiento de diversas tareas de clasificación, tales como 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 un proceso lento y altamente demandante de recursos computacionales. Afortunadamente, existen redes pre-entrenadas en base a grandes volumenes de datos, las cuales pueden ser usadas para obtener un vector de caraterísticas de alta discriminatividad. En el caso de imágenes, la red neuronal profunda conocida como Alex's Net, que pronto veremos en clases, es un ejemplo de esto, y será la herramienta usada para obtener un descriptor (feature vector) de cada una de las imágenes usadas en esta actividad.

2.2 Preprocesamiento: Ventana deslizante y max-pooling

Una de las características relevantes del mundo visual es la coherencia local de diversos 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.

Las imágenes usadas en esta actividad tienen una resolución de 256x320 pixeles, por tanto, al utilizar una ventana deslizante 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 (feature vector) de 4096 dimensiones para 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 dimensión 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 este laboratorio, para cada imagen el proceso de max-pooling entrega un vector de 4096 dimensiones.

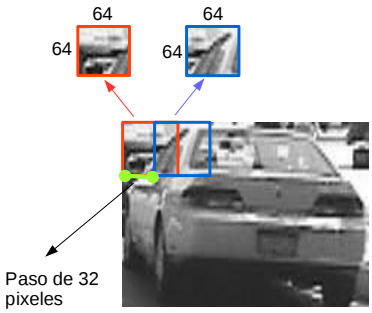

Figure 1: Ventana deslizante usando patches de 64x64 pixeles y un paso de 32 pixeles.

3 Plataforma de Software

La actividad la realizaremos utilizando códigos en lenguage de programación Python. Una de las grandes ventajas de usar Python es la gran cantidad de librerías disponibles con código abierto, lo cual ofrece gran flexibilidad para utilizar diversos tipos de datos, variadas técnicas de aprendizaje de máquina y diferentes métodos de visualización. En particular, en el ámbito del manejo efectivo de datos, o data science, la plataforma Anaconda (https://www.continuum.io/) permite una fácil instalación, organización y manejo de librerías para Python. Adicionalmente, la librería scikit-learn http:\www.scikit-learn.org provee una gran gama de técnicas de aprendizaje de máquina. Estas herramientas serán la base para el desarrollo de esta actividad práctica.

3.1 Instalación de Anaconda

La actividad la podrán realizar en Linux o MS-Windows. En el siguiente link podrán descargar Anaconda para estos sistemas operativos, así como instrucciones de instalación: https://www.continuum.io/downloads. **Importante:** deben descargar e instalar la version para Python 2.7.

Si bien Anaconda contiene versiones recientes de las librerías más populares para análisis de datos en Python, en el caso de scikit-learn sólo una versión muy reciente incorporó un clasificador basado en redes neuronales, por tanto, para utilizar esta nueva potencialidad es necesario realizar la siguiente actualización:

- Abran un terminal. En el caso de linux pueden ir al menú o ejecutar control+alt+T. En el caso de Windows vayan al menú y seleccionen la pestaña "Terminal" o "Anaconda Prompt" (el rótulo puede variar dependiendo si usan la versión en inglés o español).
- En el terminal ejecuten: *conda update scikit-learn*

3.2 Códigos y set de datos

Desde el sitio web del curso (moodle), sección "Actividades de Laboratorio", descarguen el siguiente material:

- Datasets Lab 1: MIT-10-Classes.zip y MIT-67-Classes.zip.
- Códigos Lab 1 Parte 1: archivos con código Python para ejecutar la actividad.

4 Actividades

Las siguientes actividades consistirán en ejecutar diversos códigos de Python. Si bien no tendrán que programar, si deberán entender los principales pasos de cada código, así como modificar algunos parámetros para probar técnicas vistas en clases. Para ejecutar los códigos pueden utilizar el IDE (integrated development environment) que más les acomode, sin embargo, recomendamos usar Spyder, que es el IDE incorporado en Anaconda. Para ejecutar Spyder:

- Abran un terminal, luego ejecuten: *spyder*.

4.1 MIT-67: 10 clases

Comenzaremos probando las técnicas vistas en clases en un set reducido de 10 categorías de escenas del set MIT-67. El usar un set de datos reducido facilitará el análisis y permitirá acelerar el tiempo de entrenamiento.

Para esta actividad usaremos el código en el archivo Actividad1.py. Antes de comenzar, debemos verificar que las rutas locales sean correctas. Para ello sigan los siguientes pasos:

- Abran el archivo Actividad1.py. En la línea 3 escriban la ruta donde se ubican los archivos para esta actividad. Por ejemplo, si Actividad1.py está ubicado en `C:\BigData\Lab1`, escriban: sys.path.append(`'C:\BigData\Lab1'`). En linux utilice slash (`/`) en lugar de backslash (`\`), i.e., escriban: sys.path.append(`'/BigData/Lab1'`).
- Descompriman los archivos de datos dentro del mismo directorio donde se encuentran los archivos con código Python. De lo contrario, deberán modificar el archivo initOptions.py indicando la ruta del directorio donde se encuentran los datasets.

4.2 Vecinos Cercanos

Ya estamos listos para comenzar a aplicar los clasificadores a evaluar: vecinos cercanos (KNN), máquinas de vectores de soporte (SVM) y redes neuronales (NNS). La selección del clasificador a ejecutar se realiza en la sección 4 del archivo Actividad1.py, seleccionando: KNN, SVM, o NNS, según corresponda.

- Utilizando el set MIT-10-Classes pruebe el rendimiento de un clasificador de vecinos cercanos. Para el clasificador utilice distancia euclideana, 1 vecino cercano y **no** pondere el voto de los vecinos según distancia. **Obs:** para ponderar el voto en forma inversamente proporcional a la distancia se debe utilizar *weights = 'distance'*, para no ponderar se debe utilizar *weights = 'uniform'*.
- Documente sus resultados reportando el nivel de exactitud alcanzado para el set de entrenamiento y test. ¿Nota diferencias entre los rendimientos obtenidos en estos sets?, ¿Por qué el rendimento en el set de entrenamiento es tan alto?, comente.
- Modifique el número de vecinos a 5 y vuelva a probar el rendimiento del clasificador. ¿Nota diferencias entre los rendimientos obtenidos?, ¿Por qué ahora el rendimiento en el set de entrenamiento no es 100%?. Fundamente sus respuestas.
- Analice la matriz de confusión reportada por el programa, ¿Cuál es el error más común?, ¿Qué categoría obtiene el mejor rendimiento?, ¿Cuál es el más bajo?. Indique razones que puedan justificar estos resultados. Para esto mire las imágenes reales incluidas en el set de datos.
- Indique 3 acciones que podría intentar para generar un mejor clasificador de vecinos cercanos para el set MIT-10-Classes. Intente al menos una de ellas.

4.3 Máquinas de Vectores de Soporte

Utilizando el set MIT-10-Classes pruebe el rendimiento de un clasificador SVM. Para el clasificador utilice los valores por defecto.

- Documente sus resultados utilizando el nivel de exactitud reportado para el set de entrenamiento y test. ¿Nota diferencias entre los rendimientos obtenidos para estos sets?, comente.
- Analice la matriz de confusión reportada, ¿Cuál es el error más común?, ¿Qué categoría obtiene el mejor rendimiento?, ¿Cuál es el más bajo?. Indique razones que puedan justificar estos resultados. Para esto mire las imágenes reales incluidas en el set de datos.
- Vuelva a ejecutar el clasificar utilizando distintos valores para el coeficiente de penalización de las variables slack. Para ello, en la definición del clasificador modifique el valor de C (por defecto *C=1.0*).

4.4 Redes Neuronales

Utilizando el set MIT-10-Classes pruebe el rendimiento de una red neuronal. Para el clasificador utilice los siguientes valores:

- Dos capas ocultas de 100 unidades cada una: *hidden_layer_sizes=(100, 100)*.
- Para la optimización use sthocastic gradient descent: *solver=sgd*.
- Para la tasa de aprendizaje use un valor constante de 0.001: *learning_rate='constant'*, *learning_rate_init=0.001*.

Repita el análisis de resultados realizado para los clasificadores anteriores:

- Reporte el nivel de exactitud en set de train y test, así como un análisis de la matriz de confusión.
- Modifique el número de unidades en las capas ocultas, comente sus observaciones.
- Modifique el método para realizar la optimización, pruebe con las siguientes opciones y comente sus resultados:
  - lbfgs (*solver='lbfgs'*): es un optimizador basado en derivadas de primer y segundo orden.
  - adam (*solver='adam'*): es un optimizador de primer orden similar al gradiente estocástico pero maneja en forma activa la tasa de aprendizaje de cada peso de la red.

4.5 MIT-67: 67 clases

En esta parte de la actividad usaremos el set completo de 67 clases. Nuevamente evaluaremos los clasificadores de vecinos cercanos, máquinas de vectores de soporte y redes neuronales. Para ello usaremos el archivo Actividad2.py.

- Compare la operación y resultados de los 3 clasificadores bajo evaluación. En particular, realice una tabla comparativa que considere los siguientes aspectos:
  - Facilidad de uso.
  - Tiempos de proceso en términos de entrenamiento y test.
  - Exactitud de la clasificación.
  - Tipos de error.
- Compare los rendimientos obtenidos para los sets MIT-10-Classes y MIT-67-Classes. ¿ Por qué en el set MIT-67 baja considerablemente el rendimiento en el set de test, pese a que este set tiene más datos?
- Indique 2 posibles acciones que podrían mejorar el rendimiento de clasificación en el set MIT-67-Classes.