Métodos Numéricos
Primer Cuatrimestre 2015
Trabajo Práctico 2

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Si nos organizamos aprobamos todos...

Introducción

Uno de los momentos de mayor tensión durante un cuatrimestre se genera en épocas de parciales. El día del examen, luego de entregar la resolución del mismo, suele aparecer esa inmanejable incertidumbre en los alumnos: *¿entregué todas las hojas?*, *¿consideré el caso particular de* $\beta = 0$*?*, *¿por qué mi compañero puso verdadero, si yo encontré un contraejemplo?* Desde el lado del docente, la dificultad radica en corregir un gran volumen de ejercicios técnicamente complejos en un tiempo acotado, buscando deducir en base a información limitada si el alumno comprendió o no el tema.

Estos dos factores llevaron a la conformación de una comisión interclaustro de docentes y alumnos, para quienes resguardamos su identidad, a proponer como solución el desarrollo de una herramienta automática de corrección de exámenes, con el fin de reducir los tiempos en la devolución de los mismos. Para evitar suspicacias respecto de las correcciones, se propuso dividir el problema en varias etapas y que el sistema sea desarrollado por alumnos, contando desde ya con la validación de los docentes. La primera etapa del proyecto consiste en desarrollar una herramienta que permita automáticamente identificar dígitos manuscritos, a ser aplicada a una digitalización de cada examen, y cuyo post-procesamiento involucrará darle semántica y reglas a la información obtenida.

Como es de esperar, el éxito de la herramienta depende inevitablemente de disponer de un buen reconocedor de dígitos manuscritos. Para semejante responsabilidad, dado el gran nivel mostrado durante el comienzo del cuatrimestre, la comisión asesora ha decidido elegir a los alumnos de Métodos Numéricos para llevar adelante el desarrollo.

Objetivos y Metodología

Como instancias de entrenamiento, se tiene una base de datos de n imágenes de $M \times M$ píxeles, las cuales se encuentran etiquetadas con el dígito, 0-9, al que corresponden. Definimos como $m = M \times M$ al número total de píxeles de una imagen. Asumiremos también que las imágenes se encuentran en escala de grises (cada pixel corresponde a un valor entre 0 y 255, indicando la intensidad del mismo) y que el etiquetado no contiene errores. El objetivo del trabajo consiste en utilizar la información de la base de datos para, dada una nueva imagen de un dígito sin etiquetar, determinar a cuál corresponde teniendo en cuenta factores de calidad y tiempo de ejecución requeridos.

Una primera aproximación es utilizar el conocido algoritmo de k *Vecinos más Cercanos* (kNN, por su nombre en inglés). En su versión más simple, este algoritmo considera a cada objeto de la base de entrenamiento como un punto en el espacio, para el cual se conoce a qué clase corresponde (en nuestro caso, qué dígito es). Luego, al obtener un nuevo objeto que se busca clasificar simplemente se buscan los k vecinos más cercanos y se le asigna la clase que posea el mayor número de repeticiones dentro de ese subconjunto, es decir, la moda. Con este objetivo, podemos representar a cada imagen de nuestra base de datos como un vector $x_i \in \mathbb{R}^m$, con $i = 1, \ldots, n$, y de forma análoga interpretar las imágenes a clasificar mediante el algoritmo

kNN.

Sin embargo, el algoritmo del vecino más cercano es sensible a la dimensión de los objetos a considerar y, aún aplicando técnicas de preprocesamiento, puede resultar muy costoso de computar. Teniendo en cuenta esto, una alternativa interesante de preprocesamiento que busca reducir el tamaño de los elementos a considerar es el método de *Análisis de Componentes Principales* (PCA, por su sigla en inglés), que consiste en lo siguiente. Para $i = 1, \ldots, n$, recordar que $x_i \in \mathbb{R}^m$ la i-ésima imagen de nuestra base de datos almacenada por filas en un vector, y sea $\mu = (x_1 + \ldots + x_n)/n$ el promedio de las imágenes. Definimos $X \in \mathbb{R}^{n \times m}$ como la matriz que contiene en la i-ésima fila al vector $(x_i - \mu)^t/\sqrt{n-1}$, y

$$A = U\Sigma V^t$$

a su descomposición en valores singulares, con $U \in \mathbb{R}^{n \times n}$ y $V \in \mathbb{R}^{m \times m}$ matrices ortogonales, y $\Sigma \in \mathbb{R}^{n \times m}$ la matriz diagonal conteniendo en la posición (i, i) al i-ésimo valor singular σ_i. Siendo v_i la columna i de V, definimos para $i = 1, \ldots, n$ la *transformación característica* del dígito x_i como el vector $\mathbf{tc}(x_i) = (v_1^t\, x_i, v_2^t\, x_i, \ldots, v_\alpha^t\, x_i) \in \mathbb{R}^\alpha$, donde $\alpha \in \{1, \ldots, m\}$ es un parámetro de la implementación. Este proceso corresponde a extraer las α primeras *componentes principales* de cada imagen. La intención es que $\mathbf{tc}(x_i)$ resuma la información más relevante de la imagen, descartando los detalles o las zonas que no aportan rasgos distintivos. Dada una nueva imagen x de un dígito, se calcula $\mathbf{tc}(x)$ y se compara con $\mathbf{tc}(x_i)$, para $i = 1, \ldots, n$, utilizando algún criterio de clasificación adecuado, como por ejemplo kNN.

Finalmente, nos concentramos en la evaluación de los métodos. Dado que necesitamos conocer a qué dígito corresponde una imagen para poder verificar la correctitud de la clasificación, una alternativa es particionar la base de entrenamiento en dos, utilizando una parte de ella en forma completa para el entrenamiento y la restante como test, pudiendo así corroborar la predicción realizada. Sin embargo, realizar toda la experimentación sobre una única partición de la base podría resultar en una incorrecta estimación de parámetros, como por ejemplo el conocido *overfitting*. Luego, se considera la técnica de *cross validation*, en particular el *K-fold cross validation*, para realizar una estimación de los parámetros del modelo que resulte estadísticamente más robusta.

Enunciado

Se pide implementar un programa en `C` o `C++` que lea desde archivos las imágenes de entrenamiento correspondientes a distintos dígitos y que, utilizando los métodos descriptos en la sección anterior, dada una nueva imagen de un dígito determine a qué número pertenece. Para ello, el programa deberá implementar el algoritmo de kNN así como también la reducción de dimensión previa utilizando PCA.

Con el objetivo de obtener la descomposición en valores singulares, se deberá implementar el método de la potencia con deflación para la estimación de autovalores/autovectores de la matriz de covarianza. En este contexto, la factibilidad de aplicar este método es particularmente sensible al tamaño de las imágenes de la base de datos. Por ejemplo, considerar imágenes en escala de grises de 100×100 píxeles implicaría trabajar con matrices de tamaño 10000×10000. Se pide tener en cuenta este factor durante el desarrollo y analizar cómo afecta (si es que lo hace) en el desarrollo del trabajo.

Consideramos la base de datos MNIST, en la versión disponible en *Kaggle*. Esta base contiene un conjunto de datos de entrenamiento de 42.000 dígitos manuscritos en escala de grises y con $M = 28$. A su vez, provee un conjunto de datos de test de 28.000 dígitos, de similares

características, pero sin el correspondiente etiquetado. En base a estos datos, se pide separar el procedimiento en dos etapas.

La primera de ellas consiste en realizar un estudio experimental con los métodos propuestos sobre la base de entrenamiento y utilizando la técnica de *K-fold cross validation* mencionada anteriormente. Para esta última tarea en particular, se recomienda leer y utilizar la rutina `cvpartition` provista por MATLAB. Llamamos k a la cantidad de vecinos a considerar en el algoritmo kNN, α a la cantidad de componentes principales a tomar y K a la cantidad de particiones consideradas para el cross-validation. La calidad de los resultados obtenidos serán analizados considerando la tasa de efectivdad lograda, es decir, la cantidad dígitos correctamente clasificados respecto a la cantidad total de casos analizados. En función de la experimentación se piden como mínimo los siguientes experimentos:

- Analizar el comportamiento y la factibilidad de aplicar el algoritmo kNN para un rango razonable de valores distintos de k, analizando el compromiso entre el tiempo de ejecución y la calidad de los resultados obtenidos. Es posible, como experimento opcional, considerar alguna técnica alternativa de preprocesamiento de la información en caso de ser necesario (y, desde ya, que no sea PCA).
- Analizar la calidad de los resultados obtenidos al combinar PCA con kNN, para un rango amplio de combinaciones de valores de k y α. Considerar en el análisis también el tiempo de ejecución.
- Realizar los experimentos de los items anteriores para al menos dos valores distintos de K. Justificar el por qué de la elección de los mismos.
- En base a los resultados obtenidos para ambos métodos, seleccionar aquella combinación de parámetros que se considere la mejor alternativa, con su correspondiente justificación, y compararlas entre sí y sugerir un método para su utilización en la práctica.

Finalmente, y con los parámetros seleccionados en la fase experimental, ejecutar el método seleccionado sobre el conjunto de datos de test, utilizando como entrenamiento los 42.000 dígitos comprendidos en el conjunto de entrenamiento. Analizar el tiempo de cómputo requerido. Dado que la cátedra no posee la información sobre a qué dígito corresponde cada imagen (y la idea no es graficar uno por uno y obtenerlo a mano), se sugiere a cada grupo participar en la competencia *Digit Recognizer* actualmente activa en *Kaggle* realizando el/los envíos que consideren apropiados y reportar en el informe los resultados obtenidos.

Puntos opcionales (no obligatorios)

- Mostrar que si tenemos la descomposición $A = U\Sigma V^t$, V es la misma matriz que obtenemos al diagonlizar la matriz de covarianzas.
- Realizar experimentos utilizando dígitos manuscritos creados por el grupo, manteniendo el formato propuesto.[1] Reportar resultados y dificultades encontradas.
- Implmenentar alguna mejora al algoritmo de kNN.

Programa y formato de archivos

[1] Notar que en la base original las figuras están rotadas y es blanco sobre negro, y no al revés.

Se deberán entregar los archivos fuentes que contengan la resolución del trabajo práctico. El ejecutable tomará tres parámetros por línea de comando, que serán el nombre del archivo de entrada, el nombre del archivo de salida, y el método a ejectutar (0: kNN, 1: PCA + kNN).

Asumimos que la base de datos de imágenes de entrenamiento se llama `train.csv` y que la base de test `test.csv`, y que ambas siguen el formato establecido por la competencia. Para facilitar la experimentación, el archivo de entrada con la descripción del experimento tendrá la siguiente estructura:

- La primera línea contendrá el path a la(s) base(s) de datos, k, α y K.
- Luego, habra K líneas de 42.000 valores, uno por cada muestra de la base de entrenamiento, donde un 1 indicará que esa muestra se considera parte del entrenamiento, y un 0 que se considera parte del test. Luego, de esta forma se pueden codificar las particiones realizadas por el *K-fold cross validation*.

El archivo de salida obligatorio tendrá para cada partición que figure en el archivo de entrada el vector solución con los α valores singulares de mayor magnitud, con una componente del mismo por línea. Además, el programa deberá generar un archivo, cuyo formato queda a criterio del grupo, indicando la tasa de reconocimiento obtenida para cada partición. Adicionalmente, se generará un archivo que concatene la extension `.csv` al segundo valor recibido como parámetro del programa, que escribirá la predicción realizada sobre la base de test en el formato requerido por la competencia siguiendo el formato establecido por las reglas de la competencia.

Junto con el presente enunciado, se adjunta una serie de scripts hechos en `python` y un conjunto instancias de test que deberán ser utilizados para la compilación y un testeo básico de la implementación. Se recomienda leer el archivo `README.txt` con el detalle sobre su utilización.

Sobre la entrega

- Formato electrónico: Jueves 14 de Mayo de 2015, hasta las 23:59 hs., enviando el trabajo (`informe` + `código`) a `metnum.lab@gmail.com`. El `asunto` del email debe comenzar con el texto `[TP2]` seguido de la lista de apellidos de los integrantes del grupo. Ejemplo: `[TP2] Acevedo, Miranda, Montero`
- Formato físico: Viernes 15 de Mayo de 2015, de 17:30 a 18:00 hs.