Métodos Numéricos
Primer Cuatrimestre 2016
Trabajo Práctico 2

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

CSI:DC

Introducción

En la actualidad, las ciencias de la computación son útiles para resolver un abanico muy abarcativo de problemas. En los últimos años la inmersión de herramientas computacionales para la resolución de problemas de índole criminalística ha sido abrumadora, seguramente motivada por la interminable lista de películas y series *hollywoodenses* al respecto. Como muchas otras modas, esta también desembarcó en Argentina y son varios los interesados en manejar estas tecnologías para mostrarse como líderes del segmento.

En este contexto, una entidad que mantendremos en el anonimato ha mostrado interés en que un grupo selecto de personas relacionadas con el DC desarrolle un sistema que pueda reconocer texto manuscrito. Sin embargo, para lograr entablar relaciones con dicha entidad, se ha acordado entre ambas partes la entrega de un módulo de reconocimiento de dígitos como prueba de concepto. El desarrollo de dicho módulo es de extrema importancia dado que es el puntapié inicial para comenzar las relaciones, sumado a que será una parte fundamental del software completo que se desarrolle potencialmente en un futuro.

Con plena confianza, se decidió que sean los alumnos de Métodos Numéricos quienes lleven a cabo el desarrollo del proyecto.

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 es buscar reducir la cantidad de dimensiones de las muestras para trabajar con una cantidad de

variables más acotada y, simultáneamente, buscando que las nuevas variables tengan información representativa para clasificar los objetos de la base de entrada. En esta dirección, consideraremos dos métodos de reducción de dimensiones: *Análisis de Componentes Principales* (PCA, por su sigla en inglés) y *Análisis discriminante con cuadrados mínimos parciales* (PLS-DA, por su sigla en inglés).

PCA

El método de análisis de componentes principales consiste en lo siguiente. Para $i = 1, \ldots, n$, recordar que $x_i \in \mathbb{R}^m$ corresponde a 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

$$M = X^t X$$

a la matriz de covarianza de la muestra X. Siendo v_j el autovector de M asociado al j-ésimo autovalor al ser ordenados por su valor absoluto , definimos para $i = 1, \ldots, n$ la *transformación característica* del dígito x_i como el vector $\mathbf{tc_{PCA}}(x_i) = (v_1^t\, x_i, v_2^t\, x_i, \ldots, v_\alpha^t\, x_i) \in \mathbb{R}^\alpha$, 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 $\mathbf{tc_{PCA}}(x_i)$ resuma la información más relevante de la imagen, descartando los detalles o las zonas que no aportan rasgos distintivos.

PLS-DA

En el caso de PLS-DA, la idea es similar al método de componentes principales con la diferencia de la información original que se utiliza para realizar la transformación. En este caso, en vez de utilizar solamente las imágenes de entrenamiento, se utilizan también las clases a las que pertenecen dichas imágenes para influenciar la transformación característica. Definimos X de manera análoga al algoritmo PCA. $preY \in \mathbb{R}^{n \times 10}$ como una matriz que tiene un 1 en la posición $preY_{i,j}$ si la muestra i de la base de entrenamiento corresponde al dígito $j - 1$, y -1 en el caso contrario. Por último, definimos Y como la matriz resultante de tomar $preY$, sustraer el promedio de todas las filas a cada una de ellas, y dividir por la raíz cuadrada de la cantidad de muestras menos 1. Luego, se utiliza el siguiente pseudocódigo para generar los vectores w_j.

Data: X,Y,γ
Result: $w_1 \ldots w_j$
for $i = 1 \ldots \gamma$ **do**
  definir M_i como $X^t Y Y^t X$;
  calcular w_i como el autovector asociado al mayor autovalor de M_i;
  normalizar w_i utilizando norma 2; definir t_i como $X w_i$;
  normalizar t_i utilizando norma 2; actualizar X como $X - t_i t_i^t X$;
  actualizar Y como $Y - t_i t_i^t Y$;
end

En el anterior pseudocódigo, γ es un parámetro para controlar la cantidad de dimensiones a utilizar. Una vez obtenidos los vectores w_j, la transformación característica de este método se define de forma análoga como $\mathbf{tc_{PLS}}(x_i) = (w_1^t\, x_i, w_2^t\, x_i, \ldots, w_\gamma^t\, x_i) \in \mathbb{R}^\gamma$

Como se mencionó anteriormente, los métodos PCA y PLS-DA recientemente presentados no son métodos de clasificación en sí, sino que sirven para realizar una transformación de los datos de entrada. Dada una nueva imagen x de un dígito, se calcula $\mathbf{tc_m}(x)$, siendo tc_m la

transformación característica del método que se esté utilizando, y se compara con $\mathbf{tc_m}(x_i)$, 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 tanto PCA como PLS-DA.

Con el objetivo de obtener la transformaciones características de cada método, se deberá implementar el método de la potencia con deflación para la estimación de autovalores/autovectores de la matriz de covarianza en el caso de PCA y el algoritmo de deflación de PLS-DA. En este contexto, la factibilidad de aplicar estos métodos 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*[1]. 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, γ a la cantidad de dimensiones en el método PLS-DA y K a la cantidad de particiones consideradas para el cross-validation.

La calidad de los resultados obtenidos será analizada mediante diferentes métricas. En particular, la métrica más importante que debe reportarse en los experimentos es la tasa de efectividad lograda, es decir, la cantidad dígitos correctamente clasificados respecto a la cantidad total de casos analizados.

Adicionalmente, se mencionan las siguientes métricas para su potencial uso en en el análisis de los experimentos. Se deben utilizar al menos dos de las siguientes métricas, aunque no

[1] www.kaggle.com

necesariamente para todos los experimentos realizados.

Precision: Es una medida de cuántos aciertos relativos tiene un clasificador dentro de una clase particular. Es decir, dada una clase i, la *precision* de dicha clase es $\frac{tp_i}{tp_i+fp_i}$.

En la anterior fórmula, tp_i son los *verdaderos positivos* de la clase i. Es decir, muestras que realmente pertenecían a la clase i y fueron exitosamente identificadas como tales. En contraposición, fp_i son los *falsos positivos* de la clase i. Son aquellas muestras que fueron identificadas como pertenecientes a la clase i cuando realmente no lo eran.

Luego, la *precision* en el caso de un clasificador de muchas clases, se define como el promedio de las *precision* para cada una de las clases.

Recall: Es una medida de qué tan bueno es un clasificador para, dada una clase particular, identificar correctamente a los pertenecientes a esa clase. Es decir, dada una clase i, el *recall* de dicha clase es $\frac{tp_i}{tp_i+fn_i}$.

En la anterior fórmula, fn_i son los *falsos negativos* de la clase i. Es decir, muestras que pertenecían a la clase i pero que fueron identificadas con otra clase.

Luego, el *recall* en el caso de un clasificador de muchas clases, se define como el promedio del *recall* para cada una de las clases.

F1-Score: Dado que *precision* y *recall* son dos medidas importantes que no necesariamente tienen la misma calidad para un mismo clasificador, se define la métrica F1 para medir un compromiso entre el *recall* y la *precision*. La métrica $F1$ se define como $2 * precision * recall/(precision + recall)$.

Kappa de Cohen: Es una medida para indicar cuánto concuerdan dos clasificadores sobre un mismo set de datos. Dicha medida se define como $\kappa = (p_o - p_a)/(1 - p_a)$. Donde p_o es la probabilidad observada de que los dos clasificadores concuerden y p_a es la probabilidad aleatoria de que lo hagan.

Esta métrica puede utilizarse para determinar si el problema contiene ejemplos particularmente complicados, porque por ejemplo ningún clasificador lo reconoce correctamente.

Experimentación

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 ni PLS-DA).

- Analizar la calidad de los resultados obtenidos al combinar PCA y PLS-DA con kNN, para un rango amplio de combinaciones de valores de k, α y γ. 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, 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 $M = U\Sigma V^t$, V es la misma matriz que obtenemos al diagonalizar la matriz de covarianzas.
- Realizar experimentos utilizando dígitos manuscritos creados por el grupo, manteniendo el formato propuesto.[2] Reportar resultados y dificultades encontradas.
- Proponer y/o implementar alguna mejora al algoritmo de kNN.

Programa y formato de archivos

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 ejecutar (0: kNN, 1: PCA + kNN, 2: PLS-DA + 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, γ 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 un vector con los α autovalores de la matriz de covarianza de mayor magnitud, con una componente del mismo por línea. Seguido a este primer vector debe haber un vector con los γ autovalores de las primeras γ iteraciones de la transformación PLS-DA, nuevamente con una componente 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

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

de la implementación. Se recomienda leer el archivo `README.txt` con el detalle sobre su utilización.

Sobre la entrega

- FORMATO ELECTRÓNICO: Jueves 19 de Mayo de 2016, 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] Asad, Flores, Pompei`
- FORMATO FÍSICO: Viernes 20 de Mayo de 2016, de 17:30 a 18:00 hs.