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

Reconocimiento de dígitos

Introducción

El objetivo de este trabajo práctico es el desarrollo y evaluación de una herramienta de OCR (*Optical Character Recognition*) para el reconocimiento de dígitos manuscritos en imágenes.

Se quiere desarrollar un algoritmo de clasificación supervisado el cual se deberá *entrenar* con una base de caracteres **conocidos** que luego nos servirá para reconocer otras instancias de esos caracteres no presentes en la base de datos de entrenamiento.

El foco de este trabajo está puesto en la experimentación científica de un método de OCR simplificado, clasificación por vecinos más cercanos con reducción de la dimensionalidad.

Metodología

En este trabajo práctico nos limitaremos a reconocer dígitos manuscritos entre el 0 y el 9. Sin embargo, es deseable que este método pueda ser extendido para reconocer cualquier tipo de caracteres o símbolos.

Como instancias de entrenamiento, se tiene una base de datos de n imágenes en escala de grises, las cuales se encuentran etiquetadas con su clase correspondiente, es decir, el dígito, 0-9, al que representan.

Reconocimiento de dígitos

El objetivo del reconocimiento de dígitos consiste en utilizar la información de la base de datos para, dada una nueva imagen de un dígito sin etiquetar, determinar a cuál corresponde.

Una primera aproximación es utilizar el conocido algoritmo de *k vecinos más cercanos* o kNN [2], por su sigla en inglés. En su versión más simple, este algoritmo considera a cada objeto de la base de entrenamiento como un punto en el espacio euclídeo m-dimensional, para el cual se conoce a qué clase corresponde (en nuestro caso, qué dígito es) para luego, dado un nuevo objeto, asociarle la clase del o los puntos más cercanos de la base de datos.

Procedimiento de k vecinos más cercanos

- Se define una base de datos de entrenamiento como el conjunto $\mathcal{D} = \{x_i \ : \ i = 1, \ldots, n\}$.
- Luego, se define m como el número total de píxeles de la imagen i-ésima almacenada por filas y representada como un vector $x_i \in \mathbb{R}^m$.
- De esta forma, dada una nueva imagen $x \in \mathbb{R}^m$, talque $x \notin \mathcal{D}$, para clasificarla simplemente se busca el subconjunto de los k vectores $\{x_i\} \subseteq \mathcal{D}$ más cercanos a x, y se le asigna la clase que posea el mayor número de repeticiones dentro de ese subconjunto, es decir, la moda.

El algoritmo del vecino más cercano es muy sensible a la dimensión de los objetos y a la variación de la intensidad de las imágenes. Es por eso, que las imágenes dentro de la base de datos $\mathcal{D}$ se suelen *preprocesar* para lidiar con estos problemas.

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 el método de reducción de dimensionalidad *Análisis de componentes principales* o PCA (por su sigla en inglés) dejando de la lado los procesamientos de imágenes que se puedan realizar previamente o alternativamente a aplicar PCA.

Análisis de componentes principales

El método de análisis de componentes principales o PCA consiste en lo siguiente.

Sea $\mu = (x_1 + \ldots + x_n)/n$ el promedio de las imágenes $\mathcal{D} = \{x_i : i = 1, \ldots, n\}$ tal que $x_i \in \mathbb{R}^m$. Definimos $X \in \mathbb{R}^{n \times m}$ como la matriz que contiene en la i-ésima fila al vector $(x_i - \mu)^t/\sqrt{n-1}$. La matriz de covarianza de la muestra X se define como $M = X^t X$.

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 $\mathbf{tc}(x_i) = (v_1\, x_i, v_2\, x_i, \ldots, v_\alpha\, x_i)^t \in \mathbb{R}^\alpha$, 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 idea es que $\mathbf{tc}(x_i)$ resuma la información más relevante de la imagen, descartando los dimensiones menos significativas.

Clasificación con kNN y PCA

El método PCA previamente presentado sirve para realizar una transformación de los datos de entrada a otra base y así trabajar en otro espacio con mejores propiedades que el original. Por lo tanto, el proceso completo de clasificación se puede resumir como:

Dada una nueva imagen x de un dígito, se calcula $\mathbf{tc}(x)$, y se compara con $\mathbf{tc}(x_i)$, $\forall x_i \in \mathcal{D}$, para luego clasificar mediante kNN.

Validación cruzada

Finalmente, nos concentramos en la evaluación de los métodos y en la correcta elección de sus parámetros.

Dado que necesitamos conocer previamente a qué dígito corresponde una imagen para poder estimar 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 clasificación realizada, al contar con el etiquetado del entrenamiento. 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 caso de *overfitting*.

Por lo tanto, se estudiará la técnica de *cross validation*[2], en particular el *K-fold cross validation*[1], para realizar una estimación de los parámetros de los métodos 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 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.

Se recomienda realizar tests para verificar la implementación del método en casos donde los autovalores y autovectores sean conocidos se antemano. También, puede resultar de utilidad comparar con los datos provistos por la cátedra, utilizando Python o alguna librería de cálculo numérico.

Se pide realizar un estudio experimental de los métodos propuestos sobre una base de entrenamiento utilizando la técnica *K-fold cross validation* mencionada anteriormente, con el objetivo de analizar el poder de clasificación y encontrar los mejores parámetros de los métodos. Se deberá trabajar al menos con la base de de datos MNIST, en la versión disponible en *kaggle* para la competencia *Digit Recognizer*[2].

Experimentación

Para guiar la experimentación, se detallan los siguientes lineamientos y preguntas:

- Analizar la calidad de los resultados obtenidos al combinar kNN con y sin PCA, para un rango amplio de combinaciones de valores de k y α. Llamamos k a la cantidad de vecinos a considerar en el algoritmo kNN y α a la cantidad de componentes principales a tomar.
- Analizar la calidad de los resultados obtenidos al combinar kNN con PCA, para un rango amplio de cantidades de imágenes de entrenamiento. Utilizar desde muy pocas imágenes de entrenamiento hasta todas las disponibles para identificar en que situación se comporta mejor cada uno de los métodos.
- ¿Cómo se relaciona k con el tamaño del conjunto de entrenamiento? Pensar el valor máximo y mínimo que puede tomar k y qué sentido tendrían los valores.

[1] No confundir el K de K-fold con el k de kNN, ambos son parámetros de los métodos respectivos que no están relacionados necesariamente

[2] `www.kaggle.com/c/digit-recognizer`

También, **se debe** considerar en los análisis anteriores el tiempo de ejecución.

La calidad de los resultados obtenidos será analizada mediante diferentes métricas:

1. Accuracy
2. Curvas de precision/recall
3. Kappa de Cohen
4. F1-Score

En particular, la métrica más importante que **debe** reportarse en los experimentos es la tasa de efectividad lograda o *accuracy*, es decir, la cantidad dígitos correctamente clasificados respecto a la cantidad total de casos analizados. También, se **debe** utilizar al menos otra de las métricas mencionadas, aunque no necesariamente para todos los experimentos realizados.

- Realizar los experimentos de los ítems anteriores para valores distintos de K del método K-fold[3], donde K a la cantidad de particiones consideradas para el cross-validation.
  - Justificar el por qué de la elección de los mismos. ¿Cuál sería su valor máximo?
  - ¿En qué situaciones es más conveniente utilizar K-fold con respecto a no utilizarlo?
  - ¿Cómo afecta el tamaño del conjunto de entrenamiento?
- 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.

En todos los casos es **obligatorio** fundamentar los experimentos planteados, proveer los archivos e información necesaria para replicarlos, presentar los resultados de forma conveniente y clara, y analizar los mismos con el nivel de detalle apropiado. En caso de ser necesario, es posible también generar instancias artificiales con el fin de ejemplificar y mostrar un comportamiento determinado.

Puntos opcionales (no obligatorios)

La factibilidad de aplicar PCA es particularmente sensible al tamaño de las imágenes de la base de datos. Por ejemplo, al considerar imágenes en escala de grises de 100×100 píxeles implica trabajar con matrices de tamaño 10000×10000. Tener en cuenta este factor durante el trabajo práctico y analizar cómo afecta (si es que lo hace) en el desarrollo del trabajo.

- Mostrar que si tenemos la descomposición $M = U\Sigma V^t$, V es la misma matriz que obtenemos al diagonalizar la matriz de covarianzas.

[3]Para esta tarea en particular, se recomienda leer la rutina `cvpartition` provista por MATLAB.

- Realizar experimentos utilizando dígitos manuscritos creados por el grupo, u otro tipo de caracteres, manteniendo el formato propuesto.[4]. Tener en cuento lo mencionado sobre el tamaño de las matrices a procesar con PCA. Reportar resultados y dificultades encontradas.
- Proponer y/o implementar alguna mejora al algoritmo de kNN.
- $\star$ Con los mejores parámetros seleccionados en la fase experimental, ejecutar el método sobre el conjunto de datos de test `test.csv`, utilizando como entrenamiento los 42.000 dígitos de comprendidos en el conjunto de entrenamiento `train.csv`.
  Dado que la cátedra no posee la información sobre a qué dígito corresponde cada imagen de `test.csv` (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.

Formato de entrada/salida

El ejecutable producido por el código fuente entregado deberá contar con las funcionalidades pedidas en este apartado. El mismo deberá tomar al menos tres parámetros por línea de comando con la siguiente convención:

```
$ ./tp2 -m <method> -i <train_set> -q <test_set> -o <classif>
```

donde:

- `<method>` el método a ejecutar con posibilidad de extensión (0: kNN, 1: PCA + kNN, ... etc)
- `<train_set>` será el nombre del archivo de entrada con los datos de entrenamiento.
- `<test_set>` será el nombre del archivo con los datos de test a clasificar
- `<classif>` el nombre del archivo de salida con la clasificación de los datos de test de `<test_set>`

Todos los archivos de entrada/salida deberán estar en `.csv` y siguiendo el formato establecido por la competencia de *kaggle*. Un ejemplo de invocación con los datos provistos por la cátedra sería el siguiente:

```
$ ./tp2 -m 1 -i train.csv -q test.csv -o sample_submission.csv
```

Además, el programa deberá imprimir por consola un archivo, cuyo formato queda a criterio del grupo, indicando la tasa de reconocimiento obtenida para cada conjunto de test y los parámetros utilizados para los métodos.

Nota: cada grupo tendrá la libertad de extender las funcionalidades provistas por su ejecutable. En particular, puede ser de utilidad alguna variante de toma de parámetros que

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

permita entrenar con un porcentaje de la base de datos de entrenamiento y testear con el resto (ver archivos provistos por la cátedra). Además, puede ser conveniente separar la fase de entrenamiento de la de testeo/consulta para agilizar los cálculos.

Fecha de entrega

- Formato Electrónico: Domingo 1 de Noviembre de 2020, hasta las 23:59 hs, enviando el trabajo (informe + código) a la dirección `metnum.lab@gmail.com`. El subject del email debe comenzar con el texto `[TP2]` seguido de la lista de apellidos de los integrantes del grupo.

Importante: El horario es estricto. Los correos recibidos después de la hora indicada no serán considerados.

Referencias

[1] Tinku Acharya and Ajoy K Ray. *Image processing: principles and applications*. John Wiley & Sons, 2005.

[2] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.