Universidad de
SanAndrés

I302 - Aprendizaje Automático y Aprendizaje Profundo

1er Semestre 2024

Trabajo Práctico 4

Fecha de entrega: Viernes 31 de mayo, 23:59 hs.

Formato de entrega: Los archivos desarrollados deben ser entregados en un archivo comprimido .zip a través del Campus Virtual, utilizando el siguiente formato de nombre de archivo: *Apellido_Nombre_TPX.zip*, donde X es el número de TP. Se aceptará únicamente 1 archivo por estudiante. En caso de que el nombre del archivo no cumpla con la nomenclatura especificada, el trabajo no será corregido.

La carpeta comprimida deberá constar de N sub-carpetas, una por cada problema del TP (es decir, cada problema tiene su sub-carpeta denominada "Problema N"). Dentro de cada sub-carpeta deberán incluir un Jupyter Notebook (archivo *.ipynb*) en el cual se den las respuestas a los incisos del problema y se muestren los gráficos resultantes. Pueden agregar en el Jupyter Notebook mas resultados o análisis de los expresamente solicitados en los incisos, si lo consideran adecuado. Junto con el Jupyter Notebook podrá haber uno o mas archivos de Python que contengan código que sea parte de la resolución del problema. Dicho código debe seguir las buenas prácticas de programación y modularización que se presentaron en clase.

---

1. **Clustering de datos.** Para el dataset *clustering.csv* realizar los siguientes análisis:

   (a) Implementar el algoritmo K-means y determinar la cantidad de clusters con el método de "ganancias decrecientes" (graficar L vs. K, y elegir un valor K donde al aumentar K deje de reducir significativamente L). Graficar el conjunto de datos x_i mostrando a qué cluster pertenece cada dato (usando colores/marcadores distintos para cada cluster) y también mostrar el centroide de cada cluster.

   (b) Implementar el algoritmo Gaussian Mixture Model (GMM) y realizar la misma tarea que en el inciso anterior. Recuerde que puede inicializar la optimización de GMM con una corrida de K-means.

   (c) Implementar el algoritmo DBSCAN y aplicarlo al conjunto de datos. Explorar el efecto de variar los parámetros ϵ (radio de la vecindad) y K (mínimo número de puntos en una zona densa). Luego, elegir una combinación razonable de ϵ y K y graficar los datos mostrando a qué cluster pertenece cada uno, utilizando colores/marcadores distintos para cada cluster/ruido.

2. **Reducción de dimensionalidad.** Este problema se basará en el dataset *MNIST_dataset.csv*, que contiene representaciones tabulares de imágenes de dígitos del 0 al 9. Originalmente, cada imagen tiene una resolución de 28x28 píxeles en escala de grises. En este conjunto de datos, cada imagen se representa como una fila de 784 (28x28) valores, donde cada valor representa la intensidad de un píxel en la imagen.

   (a) Implementar Principal Component Analysis (PCA) y aplicarlo al conjunto de datos. Graficar cómo varía el error cuadrático medio de reconstrucción sobre el conjunto de datos en función de la cantidad de componentes principales utilizadas.

   (b) Seleccionar la cantidad de componentes que considere adecuada y justifique la elección. Usando dicha cantidad de componentes, graficar las imagenes de los digitos originales y reconstruidos para las primeras 10 muestras del dataset.

   (c) OPCIONAL: Construir un modelo de autoencoder variacional (VAE) en PyTorch para procesar el conjunto de datos MNIST. Recuerde dividirlo en dos subconjuntos: entrenamiento y validación. El subconjunto de entrenamiento se empleará para entrenar el VAE, mientras que el de validación servirá para ajustar los hiperparámetros y evaluar el error de reconstrucción. Una vez desarrollado el VAE, compare la calidad de las imágenes reconstruidas con las obtenidas mediante PCA en el inciso anterior, utilizando 10 muestras aleatorias del conjunto de validación del VAE.

3. **Expectation-Maximization.** El algoritmo de Expectation-Maximization (EM) nos permite encontrar un máximo de la verosimilitud de los datos X, con respecto a los parametros w de un modelo de la probabilidad conjunta $p(X, Z/w)$, donde Z es una variable aleatoria latente, es decir, que no podemos medir pero suponemos que existe.

   (a) Derivar el algoritmo de EM para el caso de GMM, mostrando la función que resulta luego del E-step ($Q(w, w^o)$) y los valores optimos μ_j^*, Σ_j^* y π_j^* que resultan luego del M-step.

   (b) Demostrar que la función resultante del E-step es una cota inferior de la log-likelihood $ln(p(X/w))$ y que para el caso de GMM esta es cóncava en los parámetros μ_j, Σ_j y π_j.