Universidad Rey Juan Carlos

Práctica 3

Objetivo general: El objetivo principal de esta práctica es consolidar los conceptos fundamentales del tercer bloque de la asignatura **Fundamentos de Compresión de Datos** mediante su aplicación práctica a problemas de optimización relevantes en ciencia de datos. En este contexto, se trabajará con una variedad de problemas, como el ajuste de curvas polinomiales (*Polynomial Curve Fitting*), la selección de características mediante métodos como **Lasso**, **Ridge**, ***Elastic Net Regression*** y las máquinas de vectores de soporte (SVM). Cada uno de estos problemas se implementará en Python utilizando *solvers* como **CVXPY**, y los resultados se compararán con las soluciones obtenidas a través de funciones predefinidas de bibliotecas estándar de Python.

La práctica está dividida en dos partes, cada una centrada en un tema específico para facilitar la comprensión de los conceptos y su implementación. En cada apartado, se incluirá la formulación matemática del problema y su resolución paso a paso en código. Además, se plantearán preguntas clave que permitirán reflexionar sobre los fundamentos teóricos y garantizar la correcta interiorización de los conceptos.

Se espera que los estudiantes analicen y justifiquen con rigor técnico los resultados obtenidos, desarrollando una comprensión profunda tanto de los conceptos como de las herramientas empleadas. Este enfoque busca no solo resolver problemas específicos, sino también fortalecer el dominio de técnicas avanzadas en optimización y su aplicación práctica en el contexto de la compresión de datos y selección de características.

Práctica 3.1: Polynomial curve fitting

Objetivo: El propósito de esta práctica es aprender a formular, implementar y resolver problemas de optimización utilizando *solvers* predefinidos como **CVXPY**. El objetivo principal es desarrollar habilidades para traducir un problema real de ciencia de datos en un problema de minimización o maximización, comprendiendo el rol de los distintos términos en la función objetivo y las restricciones.

Material: Se proporciona el archivo **práctica3.1-alumnos.ipynb**, un notebook de Jupyter que contiene una estructura básica con fragmentos de código para guiarte a lo largo de los apartados que debes completar.

Primera Parte: Ajuste Polinómico con Datos Sintéticos

En esta parte, trabajarás con un conjunto de datos sintéticos **x**, **y**, donde **x** representará el tiempo e **y** recoge los valores de una señal sinusoidal con ruido asociada a cada instante temporal en **x**. El objetivo será ajustar un polinomio de grado 5 a estos datos siguiendo estos pasos:

1. **Generación de datos sintéticos**: Genera 100 puntos considerando dos periodos de la función seno.
2. **Planteamiento matemático**: Formula (en papel) el problema de optimización para encontrar los coeficientes del polinomio que mejor ajusten los datos observados.
3. **Implementación con CVX**: Resuelve el problema formulado en **CVXPY** para obtener los coeficientes $\mathbf{w}_{cvx}$.
4. **Solución cerrada**: Encuentra matemáticamente (en papel) la solución cerrada para **w**. Implementa esta solución en Python para obtener $\mathbf{w}_{close-form}$.
5. **Uso de funciones predefinidas**: Utiliza la función `polyfit` de Python para ajustar el polinomio y obtener los coeficientes $\mathbf{w}_{polyfit}$.
6. **Comparación de coeficientes**: Compara $\mathbf{w}_{cvx}$, $\mathbf{w}_{close-form}$, $\mathbf{w}_{polyfit}$.
7. **Medición del error**: Calcula el error entre el polinomio estimado y el real. Compara los resultados entre los distintos métodos.
8. **Reducción del error**: Propón al menos dos estrategias para reducir el error entre el polinomio estimado y el real.
9. **Regularización L1 y L2**: Repite los pasos 2-5, considerando regularización **L1** y **L2**.
10. **Comparación del objetivo**: Compara el valor de la función objetivo al resolver el problema: i) Sin regularización. ii) Con regularización **L1**. iii) Con regularización **L2**.
11. **Exploración de métodos alternativos**: Investiga si existe alguna otra función en Python para realizar ajustes polinómicos y compárala con las anteriores.

Segunda Parte: Ajuste Polinómico con Datos Reales

En esta sección, se trabajará con el conjunto de datos **Ice Cream**. El objetivo será realizar un ajuste polinómico con regularización sobre datos reales y analizar el impacto de los valores de λ en el error. Los pasos son los siguientes:

1. **División de datos**: Divide los datos en conjunto de entrenamiento (80%) y prueba (20%).
2. **Ajuste con CVX**: Estima los coeficientes del polinomio de grado 5 que mejor ajusten los datos de entrenamiento, utilizando regularización **L1** y **L2**.
3. **Evolución del error en *train***: Representa gráficamente la evolución del error en entrenamiento para ambos métodos (con **L1** y **L2**) variando el valor de λ en un rango definido, por ejemplo, 17 valores $[10^{-8}, 10^{-7},\ldots, 10^{7}, 10^{8}]$.
4. **Evolución del error en *test***: Representa gráficamente la evolución del error en el conjunto de prueba para ambos métodos al variar λ en un rango definido, por ejemplo, 17 valores $[10^{-8}, 10^{-7},\ldots, 10^{7}, 10^{8}]$.

Práctica 3.2: Selección de características.

Objetivo: El propósito de esta práctica es implementar y analizar la selección de características mediante métodos de regularización **L1**, **L2** y **Elastic Net**, así como resolver problemas de clasificación utilizando máquinas de soporte vectorial (SVM). Se busca desarrollar habilidades prácticas en formulación matemática, implementación en Python y análisis de resultados.

Material: Se proporciona el notebook **practica3.2-alumnos.ipynb**, el cual contiene funciones y fragmentos de código que servirán como guía para completar los ejercicios.

Primera Parte: Selección de características.

En esta sección, se trabajará con la base de datos **Boston Housing** para explorar la selección de características utilizando distintas regularizaciones. Los pasos son los siguientes:

1. **Carga de datos**: Utiliza la base de datos disponible en el enlace: Boston Dataset.
2. **División de datos**: Divide los datos en conjuntos de entrenamiento (80%) y prueba (20%).
3. **Formulación matemática**: Plantea en papel el problema de optimización para la selección de características mediante regularización **L1**, **L2** y ***Elastic Net***.
4. **Implementación en Python**: Resuelve el problema de optimización formulado utilizando **CVXPY** para los tres métodos de regularización.
5. **Análisis de características**: Compara los coeficientes estimados por cada modelo para un valor fijo de λ. Identifica las características más relevantes y discute los resultados obtenidos.
6. **Evolución del error en *train***: Representa gráficamente la evolución del error en el conjunto de entrenamiento para los métodos **L1**, **L2** y ***Elastic Net*** al variar λ (por ejemplo, 17 valores $[10^{-8}, 10^{-7},\ldots, 10^{7}, 10^{8}]$). Analiza los resultados obtenidos.
7. **Evolución del error en *test***: Representa gráficamente la evolución del error en el conjunto de prueba para los tres métodos al variar λ. Discute las diferencias observadas entre los métodos y la influencia de λ.

Segunda Parte: SVM

En esta sección, se implementará un clasificador basado en máquina de vectores de soporte (SVM) y se analizarán sus resultados comparativamente. Se proporciona el notebook **practica3.3-alumnos.ipynb**, el cual contiene funciones y fragmentos de código que servirán como guía para completar los ejercicios.

1. **Generación de datos sintéticos**: Crea un conjunto de datos sintéticos con 50 puntos pertenecientes a dos clases diferentes.
2. **Formulación matemática**: Plantea en papel el problema de optimización para encontrar el hiperplano que maximice la separación entre las dos clases.
3. **Implementación en Python**: Resuelve el problema formulado utilizando **CVXPY** para encontrar el hiperplano óptimo.

4. **Validación con sklearn**: Compara los resultados obtenidos con la implementación propia utilizando la clase `SVC` de la librería `sklearn.svm`. Asegúrate de que los resultados coinciden o son consistentes.
5. **Análisis del margen**: Discute la relación entre los valores de la variable ξ y los puntos que caen dentro del margen del hiperplano. Relaciona este análisis con la interpretación geométrica del margen y los puntos de soporte.
6. **Selección de características con SVM en datos reales**: Cargue los datos del dataset "load_breast_cancer" y analice los valores de los coeficientes **w** en el caso de utilizar SVM con **L1** y SVM con **L2**. Comente los resultados.

Práctica 3.3: PCA vs Robust PCA.

El objetivo de esta práctica es analizar y comparar distintos métodos de reducción de dimensionalidad aplicados a datos reales, en concreto imágenes, poniendo especial énfasis en:

- El comportamiento del PCA estándar frente a ruido gaussiano y outliers.
- Las limitaciones del PCA clásico cuando los datos contienen anomalías o ruido impulsivo.
- El uso de Robust PCA (RPCA) como alternativa basada en optimización convexa para separar:
    - Una componente de bajo rango (estructura principal).
    - Una componente dispersa (outliers/anomalías).

Material: Se proporciona el notebook practica3.4-alumnos.ipynb con código para cargar y visualizar imágenes, implementación de: i) PCA estándar mediante SVD, ii) PCA usando librerías de Python (scikit-learn), iii) Robust PCA mediante optimización convexa y visualización comparativa de resultados.

1. Analizar y ejecutar el código proporcionado.
2. ¿Qué tipo de ruido se ha añadido?
3. ¿Por qué los outliers afectan a PCA?
4. ¿Existe estructura de bajo rango en la imagen?
5. ¿Cómo justificarías el número de componentes elegidos?
6. ¿Dónde aparece el ruido en la descomposición Robust PCA?
7. ¿Qué componente representa la imagen "limpia"?
8. ¿Qué ventajas y desventajas presenta cada uno de los modelos vistos?

Nota: En todas las partes de la práctica, será fundamental implementar correctamente el código, comprender los conceptos subyacentes y analizar los resultados obtenidos.