datarebel®

Bienvenido

Data Translator

datarebel

Actividad

Regresión Regularizada

Parte 1: Cargar los datos

- Cargar los datos de la diabetes desde sklearn siguiendo las instrucciones del link.
  - https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html
- Investigar qué significan los predictores.
- Elaborar un análisis exploratorio básico para comprobar si hay valores perdidos, y trazar las distribuciones univariantes y conjuntas de los predictores y el objetivo.
  - Es posible que se necesite hacer un cambio en los datos basado en la exploración.
- Utilizar visualizaciones para entender los datos.

Parte 2: Regresón Ridge (Regularización)

- La regularización Ridge es una forma de contracción: las estimaciones de los parámetros se reducen hacia cero en comparación con las estimaciones de una regresión no regularizada. La cantidad de regularización (es decir, la severidad de la contracción) se establece a través del parámetro alfa de Ridge, que necesita ser ajustado con validación cruzada.
- Dividir los datos en conjuntos de entrenamiento y prueba. El conjunto de entrenamiento se utilizará para ajustar y afinar todos los modelos, y el conjunto de pruebas se utilizará al final para comparar los modelos finales.
- Ajustar una regresión Ridge con alfa = 0,5 al conjunto de datos de entrenamiento.
- Utilizar el modelo ajustado para generar predicciones en el conjunto de datos de prueba.
- Calcular el MSE del modelo ajustado en el conjunto de prueba.
- Estimar el error de prueba y de entrenamiento de la regresión Ridge utilizando una validación cruzada 10 veces.
- Ejemplo:

```
kf = KFold(n_splits=n_folds, random_state=random_seed)
test_cv_errors, train_cv_errors = np.empty(n_folds), np.empty(n_folds)
for idx, (train, test) in enumerate(kf.split(X_train)):
# Dividir en conjuntos de entrenamiento y prueba
# Fit regresión Ridge con los datos de entrenamiento
# Elaborar predicciones
# Calcular MSE
# Almacenar MSE como un array
```

Parte 2: Regresión Ridge

- Convertir el código de validación cruzada en una función
  - def cv(X, y, base_estimator, n_folds, random_seed=154):
  - Returns:
  - train_cv_errors, test_cv_errors: tuple of arrays
- De manera que se pueda llamar como en el siguiente ejemplo:
  - train_cv_errors, test_cv_errors = cv(X_train, y_train, Ridge(alpha=0.5), n_folds=10)
- ** Hacer un clon del modelo dentro de la función antes de ajustarlo.
  - from sklearn.base import clone
  - estimator = clone(base_estimator)

Parte 2: Regresión Ridge

- Elaborar una función para cambiar el valor de alfa iniciando en cero.
  - def train_at_various_alphas(X, y, model, alphas, n_folds=10, **kwargs):
  - Returns
  - cv_errors_train, cv_errors_test: tuple of DataFrame
  - cv_errors_train = pd.DataFrame(np.empty(shape=(n_folds, len(alphas))),
  - columns=alphas)
  - cv_errors_test = pd.DataFrame(np.empty(shape=(n_folds, len(alphas))),
  - columns=alphas)
  - for alpha in alphas:
  - return cv_errors_train, cv_errors_test
- Calcular los errores de entrenamiento y de prueba para cada valor de alfa utilizando una validación cruzada 10 veces.
- De manera que se pueda llamar con:
  - train_at_various_alphas(X_train, y_train, Ridge, ridge_alphas)

Parte 2: Regresión Ridge

- Promediar las diez estimaciones de error de entrenamiento y prueba para cada alfa con el objetivo de obtener una estimación más estable del error de entrenamiento y prueba para cada valor del parámetro de regularización.
- Elaborar las curvas de MSE promedio de entrenamiento y prueba a medida que varía alfa (Probar “log(\alpha)” en el eje X para mejorar la visualización).
- Calcular el valor de alfa que produce al mínimo error de prueba del CV, y superponer una línea vertical en el valor óptimo de alfa en el gráfico de las curvas MSE.
- Ajustar una secuencia de modelos de regresión Ridge a los datos de entrenamiento para la misma secuencia de alfa que la anterior
- Graficar las trayectorias de los coeficientes como una función de log(alfa) y superponer una línea vertical en el valor óptimo de alfa elegido por la validación cruzada.

datarebel

Parte 3: Lasso

- La regresión LASSO es útil para imponer la dispersión en los coeficientes. Es decir, es preferible si creemos que muchas de las características no son en absoluto relevantes para predecir el objetivo.
- Repetir los pasos anteriores pero esta vez utilizando la clase Lasso de sklearn.
- Es necesario utilizar una secuencia diferente de valores alfa para obtener buenos resultados.

Parte 4: Comparación

- Ajustar una regresión Ridge y una regresión LASSO a su conjunto de entrenamiento utilizando los valores de alfa óptimos.
- Calcular el MSE de estos modelos utilizando la validación de prueba (esta debería ser la primera vez que se utilizan los datos de prueba).
- Para comparar, ajuste también una regresión lineal no regularizada en los datos de entrenamiento y calcule su MSE en el conjunto de prueba.
- ¿Qué modelo elegiría?
- ¿Qué pasos seguiría antes de ponerlo en producción?

datarebel

¡Gracias!

Data Translator