Школа анализа данных
Восстановление зависимостей
Домашнее задание №4

Кошман Дмитрий

Задача 1

Вывести формулу для оценки качества построения гребневой регрессии методом скользящего контроля:

$$\hat{S} = \frac{1}{l}\sum_{i=1}^{l}\left(\frac{y_i - t_i}{1 - h_{ii}}\right)^2,$$

$$\text{где } h = F(F^T F + \lambda I)^{-1} F^T,\ t = hy,\ F \in \mathbb{R}^{l\times n},\ y \in \mathbb{R}^l$$

Предсказания гребневой регрессии в точке X выглядит как $y^* = Xw^*$, где

$$w^* = \arg\min_w \left(||Fw - y||_2^2 + \lambda||w||_2^2\right); F, y - \text{обучающая выборка}$$

Выразим w^*, посчитав градиент:

$\left[D_{w^*}(|Fw - y|^2 + \lambda|w|^2)\right](h) = \left[D_{w^*}(\langle Fw - y, Fw - y\rangle + \lambda\langle w, w\rangle)\right](h) =$

$2\langle Fw^* - y, Fh\rangle + 2\lambda\langle w^*, h\rangle = 2\langle F^T(Fw^* - y) + \lambda w^*, h\rangle = \langle\nabla_w, h\rangle,$

$\nabla_w = 2(F^T(Fw^* - y) + \lambda w^*) = 0$

$(F^T F + \lambda I)w^* = F^T y$

$w^* = (F^T F + \lambda I)^{-1} F^T y$

В случае LOO кросс валидации, модель обучается на подвыборках $\hat{F}_i \in \mathbb{R}^{l-1\times n}$, $\hat{y}_i \in \mathbb{R}^{l-1}$, которые образованы поочередным выкидыванием элементов из обучающей выборки. Тогда потери на одной кросс валидации равны

$(y_i^* - y_i)^2 = (F_i w_i^* - y_i)^2 = (F_i(\hat{F}_i^T \hat{F}_i + \lambda I)^{-1}\hat{F}_i^T \hat{y}_i - y_i)^2,$

где F_i – i-ая строчка.
Заметим, что

$(\hat{F}_i^T \hat{F}_i)_{jk} = \sum_t F_{jt}^T F_{tk} - F_{ji}^T F_{ik} = (F^T F)_{jk} - (F_i^T F_i)_{jk} \Rightarrow \hat{F}_i^T \hat{F}_i = F^T F - F_i^T F_i$

$(\hat{F}_i^T \hat{y}_i)_j = (\hat{F}_i^T)_j \hat{y}_i = \sum_t F_{jt}^T y_t - F_{ji}^T y_i = (F^T y)_j - (F_i^T y_i)_j \Rightarrow \hat{F}_i^T \hat{y}_i = F^T y - F_i^T y_i$

Тогда

$(y_i^* - y_i)^2 = \big(F_i(F^TF - F_i^TF_i + \lambda I)^{-1}(F^Ty - F_i^Ty_i) - y_i\big)^2$

Выразим t_i, h_{ii}:

$h_{ii} = (F(F^TF + \lambda I)^{-1}F^T)_{ii} = F_i(F^TF + \lambda I)^{-1}F_i^T,$

$t_i = (hy)_i = h_iy = (F(F^TF + \lambda I)^{-1}F^T)_iy = F_i(F^TF + \lambda I)^{-1}F^Ty$

Пусть $A = (F^TF + \lambda I)^{-1}$, $B = (F^TF - F_i^TF_i + \lambda I)^{-1}$, тогда

$(y_i^* - y_i)^2 = \left(\frac{(1-h_{ii})(y_i^*-y_i)}{1-h_{ii}}\right)^2 = \left(\frac{(1-F_iAF_i^T)(F_iB(F^Ty-F_i^Ty_i)-y_i)}{1-h_{ii}}\right)^2 =$
$\left(\frac{F_iB(F^Ty-F_i^Ty_i)-y_i-F_iAF_i^TF_iB(F^Ty-F_i^Ty_i)+F_iAF_i^Ty_i}{1-h_{ii}}\right)^2$

Заметим, что $F_i^TF_i = A^{-1} - B^{-1}$, тогда

$(y_i^* - y_i)^2 = \left(\frac{F_iB(F^Ty-F_i^Ty_i)-y_i-F_iA(A^{-1}-B^{-1})B(F^Ty-F_i^Ty_i)+F_iAF_i^Ty_i}{1-h_{ii}}\right)^2 =$
$\left(\frac{(F_iB(F^Ty-F_i^Ty_i)-y_i-F_iB(F^Ty-F_i^Ty_i)+F_iA(F^Ty-F_i^Ty_i)+F_iAF_i^Ty_i}{1-h_{ii}}\right)^2 =$
$\left(\frac{F_iAF^Ty-y_i}{1-h_{ii}}\right)^2 = \left(\frac{t_i-y_i}{1-h_{ii}}\right)^2$

Взяв среднее по всем подвыборкам, получим исходную формулу.

Задача 2

```
import matplotlib.pyplot as plt
import numpy as np

plt.rc('figure', figsize=(10, 6))
plt.rc('font', size=12)
```

```
text_dat = "-0.3362 1.806 2.8101 0.6112 2.2959 2.8481 0.1808 0.5416 2.793 2.0963 3.5 3.9834 4.2866 5.3261 5.2257 6.
x = np.linspace(0, 2, len(y_observed))[:, None]
y_observed = np.array(list(map(float, text_dat.split())))

def f_true(x):
    return 1 + 0.8 * x + 0.6 * x**2 + 0.3 * x**3 + 0.1 * x**4 + 0.01 * x**5
```

```
def MSE(y_predict, y_true):
    return ((y_predict - y_true) ** 2).mean()

def TrainRegression(F, y, lambda_):
    I = np.eye(F.shape[1])
    w = np.linalg.inv(F.T @ F + lambda_ * I) @ F.T @ y
    return w

def CalculateLoss(F_train, y_train, F_test, y_test, lambda_):
    w = TrainRegression(F_train, y_train, lambda_)
    y_predict = F_test @ w
    return MSE(y_predict, y_test)
```

```
def LOOCVLoss(F, y, lambda_):
    loss = 0.0
    examples = F.shape[0]

    for example in range(examples):
        F_train = np.delete(F, example, axis=0)
        y_train = np.delete(y, example, axis=0)
        F_test = F[example, None]
        y_test = y[example, None]

        loss += CalculateLoss(F_train, y_train, F_test, y_test, lambda_)

    return loss / examples
```

```
def LOOCVLossFast(F, y, lambda_):
    I = np.eye(F.shape[1])
    h = F @ np.linalg.inv(F.T @ F + lambda_ * I) @ F.T
    t = h @ y
    losses = ((y - t) / (1 - h.diagonal())) ** 2
    return losses.mean()
```

```
losses = [(LOOCVLoss(x, y_observed, lambda_),
           LOOCVLossFast(x, y_observed, lambda_)) for lambda_ in np.logspace(-10, 3)]

print(f"Does formula give the same answer as manual calculations?\n", np.allclose(*zip(*losses)))
```

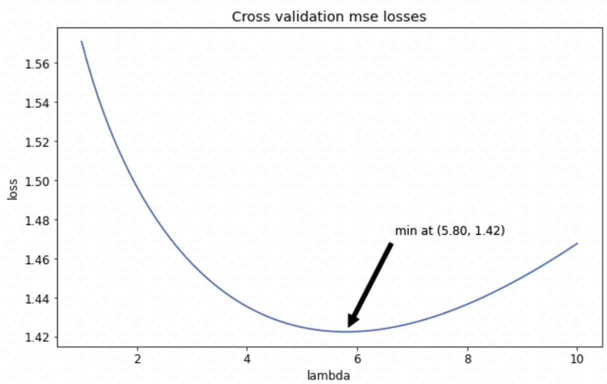
```
Does formula give the same answer as manual calculations?
 True
```

```
def GetPolynomialFeatures(F, degree):
    return np.array([F[:, 0] ** d for d in range(degree + 1)]).T

def GridSearchLambda(F, y, lambdas):
    losses = [LOOCVFast(F, y, lambda_) for lambda_ in lambdas]
    argmin = np.argmin(losses)
    x, y = lambdas[argmin], losses[argmin]

    plt.title("Cross validation mse losses")
    plt.plot(lambdas, losses)
    plt.xlabel('lambda')
    plt.ylabel('loss')
    plt.annotate(f'min at ({x:.2f}, {y:.2f})', xy=(x, y), xytext=(50, 100), textcoords='offset points',
             arrowprops=dict(facecolor='black', shrink=0.05)
             )
```

```
lambdas = np.linspace(1, 10, int(1 + (10 - 1) / .1))
F_poly = GetPolynomialFeatures(x, 5)
GridSearchLambda(F_poly, y_observed, lambdas)
```

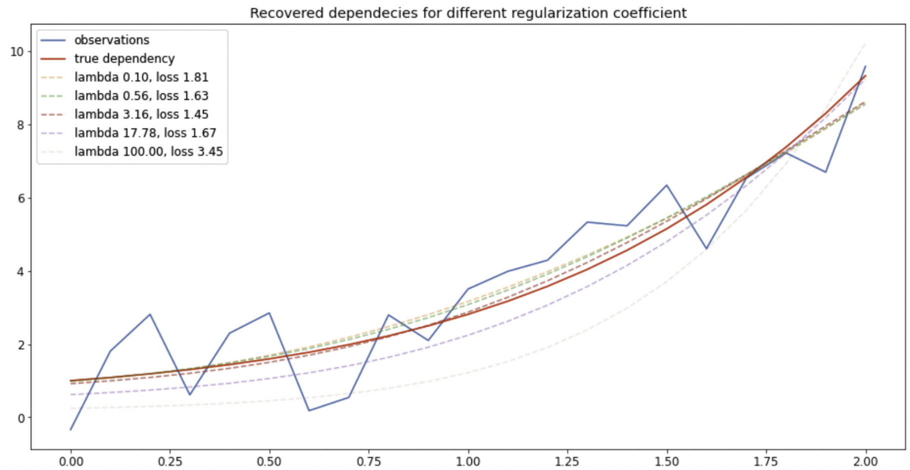



```
def PlotForLambdas(F, y, lambdas):
    plt.figure(figsize=(16, 8))
    plt.title("Recovered dependecies for different regularization coefficient")
    plt.plot(F, y, label='observations')
    plt.plot(F, [f_true(i) for i in F[:, 0]], label='true dependency', c='red')

    F_poly = GetPolynomialFeatures(F, 5)
    min_loss = min(LOOCVFast(F, y, lambda_) for lambda_ in lambdas)
    for lambda_ in lambdas:
        w = TrainRegression(F_poly, y, lambda_)
        loss = LOOCVFast(F_poly, y, lambda_)
        plt.plot(F, F_poly @ w, '--', alpha=(min_loss / loss)**2 * .7, label=f'lambda {lambda_:.2f}, loss {loss:.2f

    plt.legend()
```

```
PlotForLambdas(x, y_observed, lambdas=np.logspace(-1, 2, 5))
```



Задача 3

$y = Xa + \epsilon,\ \epsilon \sim \mathcal{N}(0, I),\ a \sim Laplace(0, I)$

$\hat{a}_{MAP} - ?$

$\hat{a}_{MAP}(X, y) = \arg\max_a p_\epsilon(\epsilon|a)p_a(a) = \arg\max_a p_\epsilon(y - Xa)p_a(a) =$

$\arg\max_a \prod_i \frac{1}{\sqrt{2\pi}} \exp\{-\frac{(y_i - X_i a)^2}{2}\} \prod_j \exp\{-\frac{|a_j|}{2}\} =$

$\arg\max_a \log \prod_{ij} \exp\{-\frac{(y_i - X_i a)^2}{2}\} \exp\{-\frac{|a_j|}{2}\} =$

$\arg\max_a \sum_{ij} -(y_i - X_i a)^2 - |a_j| =$

$\arg\min_a \sum_i (y_i - X_i a)^2 + |a_i| =$

$\arg\min_a ||y - Xa||_2^2 + ||a||_1$

Заметим, что сумма норм - выпуклая функция, значит минимум существует и единственнен. Явного выражения для $\hat{a}_{MAP}$ не существует, но можно находить его численно градиентным спуском, доопределяя градиент при попадании на координатные оси:

$\left[D_{a^*}(||y - Xa||_2^2 + ||a||_1)\right](h) = 2\langle X^T(Xa^* - y) + \text{sign}(a^*), h\rangle = \langle \nabla_a, h\rangle$

$\nabla_a = 2X^T(Xa^* - y) + \text{sign}(a^*)$

$0 = X^T X + a(X^T X)^{-1} + X^T y$

Если же $a \sim \mathcal{N}(0, I)$, то

$\hat{a}_{MAP}(X, y) = \arg\min_a ||y - Xa||_2^2 + ||a||_2^2 = (X^T X + I)^{-1} X^T y$