INF367A Exercise 6

Naphat Amundsen

February 23, 2020

Introduction

This exercise is about deriving and using regression using Bayesian methods (i.e include the prior).

1 Posterior of regression weights

Suppose $y_i = w^T x_i + \epsilon_i$ for $i = 1, \dots, n$ where $\epsilon_i \sim \mathcal{N}(0, \beta^{-1})$. Assume a prior

$$w \sim \mathcal{N}(0, \alpha^{-1} I).$$

Use 'completing the square' to show that the posterior of w is given by $P(w|y, x, \alpha, \beta) = \mathcal{N}(w|m, S)$, where

$$S = \left(\alpha I + \beta \sum_{i=1}^{n} x_i x_i^T\right)^{-1}$$

$$m = \beta S \sum_{i=1}^{n} y_i x_i$$

1) Want to get posterior (or something proportional to it) in quadratic form:

$$\frac{1}{2}\theta^T A \theta + b^T \theta \tag{1}$$

$$P(w|y, x, \alpha, \beta) \propto \overbrace{P(w)P(y, x, \alpha, \beta|w)}^{\text{Notation will be abused}} \tag{2}$$

$$= N(w|0, \alpha I) \prod_{i=1}^{n} \mathcal{N}(\overbrace{y_i - w^T x_i}^{\mathcal{N}(-x|0,\Sigma) = \mathcal{N}(x|0,\Sigma)} |0, \beta^{-1}) \tag{3}$$

$$\propto \exp\left[-\frac{1}{2} w^T \alpha I w\right] \prod_{i=1}^{n} \exp\left[-\frac{1}{2}(y_i - w^T x_i)^T \beta (y_i - w^T x_i)\right] \tag{4}$$

Logarithm the thing to make math doable

$$\Rightarrow \ln P(w|y,x,\alpha,\beta) = \ln\left(\exp\left[-\frac{1}{2}w^T\alpha I w\right]\prod_{i=1}^{n}\exp\left[-\frac{1}{2}(y_i - w^Tx_i)^T\beta(y_i - w^Tx_i)\right]\right) \tag{5}$$

$$= -\frac{1}{2}w^T\alpha I w - \frac{1}{2}\sum_{i=1}^{n}(y_i - w^Tx_i)^T\beta(y_i - w^Tx_i) \tag{6}$$

$$= -\frac{1}{2}\left[w^T\alpha I w + \sum_{i=1}^{n}(y_i - w^Tx_i)^T\beta(y_i - w^Tx_i)\right] \tag{7}$$

Let $w^Tx_i = p$ (don't want to use $\hat{y}_i$ since it confuses me when there are a lot of y_i symbols as well)

$$= -\frac{1}{2}\left[w^T\alpha I w + \sum_{i=1}^{n}(y_i - p_i)^T\beta(y_i - p_i)\right] \tag{8}$$

$$= -\frac{1}{2}\left[w^T\alpha I w + \sum_{i=1}^{n} y_i^T\beta y_i - y_i^T\beta p_i - p_i^T\beta y_i + p_i^T\beta p_i\right] \tag{9}$$

$$\propto -\frac{1}{2}\left[w^T\alpha I w + \sum_{i=1}^{n} -y_i^T\beta p_i - p_i^T\beta y_i + p_i^T\beta p_i\right] \tag{10}$$

$$= -\frac{1}{2}\Big[w^T\alpha I w + \sum_{i=1}^{n}\overbrace{-y_i^T\beta p_i - y_i^T\beta p_i + p_i^T\beta p_i}^{\beta \text{ is symmetric} \Rightarrow p_i^T\beta y_i = y_i^T\beta p_i}\Big] \tag{11}$$

$$= -\frac{1}{2}\Big[w^T\alpha I w + \sum_{i=1}^{n} -2y_i^T\beta p_i + p_i^T\beta p_i\Big] \tag{12}$$

$$= -\frac{1}{2}\Big[w^T\alpha I w + \sum_{i=1}^{n}\overbrace{-2y_i^T\beta w^Tx_i + x_i^Tw\beta w^Tx_i}^{\text{Just realized that } \beta \text{ and } y_i \text{ are in fact just scalars lol}}\Big] \tag{13}$$

$$= -\frac{1}{2}\left[w^T\alpha I w + \sum_{i=1}^{n} -2y_i^T\beta w^Tx_i + \beta w^Tx_ix_i^Tw\right] \tag{14}$$

$$= -\frac{1}{2}\left[w^T\alpha I w - 2\sum_{i=1}^{n} y_i^T\beta x_i^Tw + \sum_{i=1}^{n}\beta w^Tx_ix_i^Tw\right] \tag{15}$$

$$= -\frac{1}{2}\left[w^T\left(\alpha I + \beta\sum_{i=1}^{n}x_ix_i^T\right)w - 2\left(\beta\sum_{i=1}^{n}y_i^Tx_i^T\right)w\right] \tag{16}$$

$$= -\frac{1}{2}w^T\left(\alpha I + \beta\sum_{i=1}^{n}x_ix_i^T\right)w + \left(\beta\sum_{i=1}^{n}y_i^Tx_i^T\right)w \tag{17}$$

Now it is in quadratic form yo, let

$$A = \left(\alpha I + \beta \sum_{i=1}^{n} x_i x_i^T\right), \quad b = \left(\beta \sum_{i=1}^{n} y_i x_i\right) \tag{18}$$

Then by completing the square, we obtain the parameters S (variance), m (mean) for a Gaussian (see lecture slides).

$$S = A^{-1} = \left(\alpha I + \beta \sum_{i=1}^{n} x_i x_i^T\right)^{-1} \tag{19}$$

$$m = A^{-1} b = \beta S \sum_{i=1}^{n} y_i x_i \tag{20}$$

$$\underline{\underline{\Rightarrow P(w|y, x, \alpha, \beta) = \mathcal{N}(w|m, S)}} \tag{21}$$

2 Poisson regression with Laplace approximation

Poisson regression can be used to model count data. That is, the label y_i tells how many times some event occurs (y_i is a non-negative integer). In this exercise, we try to predict the number of cyclists crossing Brooklyn Bridge given that we know precipitation in New York. A Poisson regression model can be defined as

$$\begin{aligned} y_i|\theta &\sim \text{Poisson}(\exp(\theta^T x_i)) \\ \theta &\sim \mathcal{N}(0, \alpha^{-1} I) \end{aligned}$$

where y_i are the observed counts, x_i the related covariates, $i = 1, \ldots, N$, and θ are the regression weights. In this exercise, we approximate the posterior $P(\theta|y)$ using the Laplace approximation.

1. Derive the gradient $\nabla \log P(\theta|y)$ and the Hessian $H = \nabla\nabla \log P(\theta|y)$ needed for the Laplace approximation. Note that the posterior may look overly complicated but things get easier once you take the logarithm.

2. Load data new_york_bicycles.csv. The data points are daily observations from year 2017. The data set has two variables. The first variable (x) is the daily precipitation in New York. The second variable (y) is the number of cyclists crossing the Brooklyn Bridge (measured in hundreds of cyclists).

3. Split the data into training and test sets.

4. Approximate the posterior distribution for parameters θ given the training data using Laplace approximation. To get reasonable results, you need to use two-dimensional θ, that is, you should include an intercept term.

5. Plot the true posterior and the approximation. Plot the difference between the true and approximate posterior in a third figure. Is the approximation reasonable? Hint: Generate a grid for the parameter values (for example, using numpy's meshgrid). Compute unnormalized density on each grid point. Normalize by dividing the unnormalized densities by the sum over the whole grid. (This is not an exact normalization but it is close enough for our purposes.)

6. Estimate credible intervals for the predicted number of cyclists (Note: the interval depends on x). Plot test data with precipitation on x-axis and the number of cyclists on y-axis. Plot the mean of the predictive distribution as a function of x. Plot the 50% credible intervals.

7. Is your model well-calibrated? Does about 50% of test points lie within 50% credible interval (and similarly for 90% interval)?

2.1) Poisson distribution is:

$$P(K = k|\lambda) = \frac{\lambda^k e^{-k}}{k!} \tag{22}$$

Assuming data points are i.i.d

$$\begin{aligned}
\ln P(\theta|y) &\propto \ln P(\theta)P(y|\theta) && (23)\\
&= -\frac{1}{2}\theta^T \alpha I \theta + \sum_{i=1}^{n} \ln \frac{\lambda^\theta e^{-\lambda}}{\theta!} && (24)\\
&= -\frac{1}{2}\theta^T \alpha I \theta + \sum_{i=1}^{n} \ln \frac{\exp(\theta^T x_i y_i) e^{-\exp(\theta^T x_i)}}{y_i!} && (25)\\
&\propto -\frac{1}{2}\theta^T \alpha I \theta + \sum_{i=1}^{n} \ln \exp(\theta^T x_i y_i) e^{-\exp(\theta^T x_i)} && (26)\\
&= -\frac{1}{2}\theta^T \alpha I \theta + \sum_{i=1}^{n} \theta^T x_i y_i - \sum_{i=1}^{n} \exp(\theta^T x_i) && (27)
\end{aligned}$$

This is not a nice form (i.e no nice conjugate prior), so we have to approximate posterior using Laplace approximation (Taylor series expansion to quadratic term, assuming density function of posterior is normal). We must first and second derivative of what we have so far:

$$\begin{aligned}
\nabla \ln P(\theta|y) &\propto \frac{\partial}{\partial\theta}\left[-\frac{1}{2}\theta^T \alpha I \theta + \sum_{i=1}^{n} \theta^T x_i y_i - \sum_{i=1}^{n} \exp(\theta^T x_i)\right] && (28)\\
&= \underline{\underline{-\alpha I \theta + \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \exp(\theta^T x_i)}} && (29)
\end{aligned}$$

$$\begin{aligned}
H = \nabla\nabla \ln P(\theta|y) &= \frac{\partial^2}{\partial\theta^2} \ln P(\theta|y) && (30)\\
&\propto \frac{\partial}{\partial\theta}\left[-\alpha I \theta + \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \exp(\theta^T x_i)\right] && (31)\\
&= \underline{\underline{-\alpha I - \sum_{i=1}^{n} x_i x_i^T \exp(\theta^T x_i)}} && (32)
\end{aligned}$$

2.2, 2.3, 2.4) Shuffle and then take first 150 as train, rest as test. Laplace approximation (optimizing using Newton's method) gives ($\theta_0 = \mathbf{0}$):

$$\theta = \begin{pmatrix} -0.933 \\ 3.375 \end{pmatrix} \tag{33}$$

where last element is intercept term. Testing with different initial values for θ_0 converges to same result.

2.5 Get that Hessian inverse at θ from previous task is not positive semidefinite (it has negative values in diagonal). :(

A Task 2 code

```
import numpy as np
seed = 42069
np.random.seed(seed)
import pandas as pd
from matplotlib import pyplot as plt
from scipy.stats import multivariate_normal

def p(theta: np.ndarray, X: np.ndarray, y: np.ndarray, alpha: float=1):
    term1 = -0.5*np.linalg.multi_dot([theta.T, alpha*np.eye(len(theta)),
                                       theta])
    term2 = theta.T @ (X*y).sum(axis=0)
    term3 = -sum([np.e ** (theta.T @ x) for x in X])
    return term1 + term2 + term3

def dp(theta: np.ndarray, X: np.ndarray, y: np.ndarray, alpha: float=1):
    term1 = -(alpha * np.eye(len(theta))) @ theta
    term2 = (X*y).sum(axis=0).reshape(-1,1)
    term3 = -sum([x * np.e ** (theta.T @ x) for x in X]).reshape(-1,1)
    return term1 + term2 + term3

def ddp(theta: np.ndarray, X: np.ndarray, y: np.ndarray, alpha: float=1):
    term1 = -alpha * np.eye(len(theta))
    X = X[:,np.newaxis,:]
    term2 = -sum([(x.T@x) * np.e ** (theta.T @ x.T) for x in X])
    return term1 + term2

def laplace_iteration(theta: np.ndarray, X: np.ndarray, y: np.ndarray):
    dE = dp(theta, X, y)
    H_inv = np.linalg.inv(ddp(theta, X, y))
    return theta - H_inv@dE, dE, H_inv

def get_grid(n=100, xrange=(-10,10), yrange=(-10,10)):
    xspace = np.linspace(*xrange, n)
    yspace = np.linspace(*yrange, n)
    Xmesh, Ymesh = np.meshgrid(xspace, yspace)
    points = np.array([Xmesh.ravel(), Ymesh.ravel()]).T
    return points

def get_densitymap(func, funcargs=(), funckwargs={}, resolution=224,
                                     xrange=(-10,10), yrange=(-10,10)):
    points = get_grid(resolution, xrange, yrange)
    z = func(points, *funcargs, **funckwargs)
    return z.reshape((resolution, resolution)), points

if __name__ == '__main__':
    df = pd.read_csv('new_york_bicycles.csv', header=None).sample(frac=1,
                                          random_state=seed)
    X = np.column_stack([df.values[:,0], np.ones(len(df))])
    y = df.values[:,1]

    N_train = 150
```

```
train_slice = slice(0, N_train)
test_slice = slice(N_train, None)

X_train = X[train_slice]
X_test = X[test_slice]

y_train = y[train_slice]
y_test = y[test_slice]
y_col = y.reshape(-1,1)

def task2d(show=True):
    theta = np.array([[0,0]]).T

    for i in range(100):
        theta_new, dE, H_inv = laplace_iteration(theta, X, y_col)
        if np.allclose(theta_new, theta, rtol=1e-10, atol=1e-10):
            print(f'Converged at iteration {i}: {theta_new.ravel()}')
            break
        theta = theta_new

    return theta.ravel(), H_inv

def task2e(show=True):
    mu, Sigma = task2d(False)

    lol = dp(mu.reshape(-1,1), X, y_col)
    print(np.linalg.inv(ddp(lol.reshape(-1,1), X, y_col)))
    print(np.linalg.inv(ddp(mu.reshape(-1,1), X, y_col)))
    # print(ddp(lol.reshape(-1,1), X, y_col))
    # print(ddp(mu.reshape(-1,1), X, y_col))

    # Try the thing
    # multivariate_normal.pdf(np.array([[0,0]]), mean=mu, cov=Sigma)

    points = get_grid()

    if show:
        points = get_grid()

        z_pri = multivariate_normal.pdf(points, m0, s0)
        z_post = multivariate_normal.pdf(points, mu_n, sigma_n)

        fig, (ax1, ax2) = plt.subplots(1,2, figsize=(7,3))
        fig.suptitle('Actual vs Laplace approximation')
        ax1.set_title('Actual')
        ax1.imshow(z_pri.reshape((resolution,resolution)), extent=[8,
                                                    20, 12, 22], origin='
                                                    lower')

        ax2.set_title('Laplace')
        ax2.imshow(z_post.reshape((resolution, resolution)), extent=[8
                                                    , 20, 12, 22], origin='
```

```
                                                lower')

        plt.setp([ax1, ax2], xlabel='x', ylabel='y')

        fig.tight_layout(rect=[0, 0.03, 1, 0.95])

        maxima2d(ax2, points, z_post)
        maxima2d(ax1, points, z_pri)

        # plt.savefig('images/2c.pdf')
        plt.show()
        pass

# task2d(False)
task2e(False)
```