Name: ___________________________________ NetID: _____________

S&DS 365 / 665

Intermediate Machine Learning

Midterm Exam (Practice)

Wednesday, March 16, 2022

Complete all of the problems. You have 75 minutes to complete the exam.

The exam is closed book, computer, phone, etc. You are allowed one double-sided $8\frac{1}{2} \times 11$ sheet of paper with hand-written notes.

The following facts may (or may not) be helpful:

- If (X_1, X_2) are jointly Gaussian with distribution

$$\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \sim N\left(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} A & C \\ C^T & B \end{pmatrix} \right)$$

  then the conditional distributions are also Gaussian and given by

$$\begin{aligned} X_1 \,|\, x_2 &\sim N\left(\mu_1 + CB^{-1}(x_2 - \mu_2),\ A - CB^{-1}C^T\right) \\ X_2 \,|\, x_1 &\sim N\left(\mu_2 + C^T A^{-1}(x_1 - \mu_1),\ B - C^T A^{-1} C\right) \end{aligned}$$

- The function `np.linalg.inv` computes the inverse of a matrix.

1. ***Multinomial choice: Let 'em roll*** (10 points)

   For each of the following questions, circle the best answer.

   1.1. For the lasso, as the regularization parameter $\lambda \to \infty$ increases

   (a) the bias increases (True / False)

   (b) the variance increases (True / False)

   1.2. For kernel smoothing, as the bandwidth $h \to 0$ decreases to zero

   (a) the bias increases (True / False)

   (b) the variance increases (True / False)

   1.3. For Mercer kernel regression as the regularization parameter $\lambda \to \infty$ increases

   (a) the bias increases (True / False)

   (b) the variance increases (True / False)

   1.4. Let $F \sim DP(\alpha, F_0)$ be a draw from a Dirichlet process prior, where $F_0 = N(0, 1)$ is a standard Normal distribution. What is the value of $\mathbb{E}(F(1.96)) - \mathbb{E}(F(-1.96))$?

   (a) A random draw from a Dirichlet distribution

   (b) 0

   (c) 0.95

   (d) α

   (e) $\alpha/(1 + \alpha)$

   1.5. The following are statements regarding the "Chinese restaurant process" associated with a Dirichlet process prior $F \sim DP(\alpha, F_0)$. Circle all that apply.

   (a) It gives a way of sampling data X from the marginal without sampling F.

   (b) It gives a way of sampling a distribution F from the prior.

   (c) The probability assigned to a sequence of "customers" $X_1, \ldots, X_n$ depends on the ordering of the data.

   (d) There are generally more occupied "tables" as α increases.

   (e) It makes me hungry thinking about this right before lunch.

2. ***Sparsity and regularization:*** $1 + 2 = ?$ (15 points)

The "elastic net" combines ℓ_2 penalization (ridge) with ℓ_1 penalization (lasso). The problem is written in the following form:

$$\widehat{\beta} = \operatorname*{argmin}_{\beta} \frac{1}{2n} \|Y - X\beta\|^2 + \rho\lambda\|\beta\|_1 + (1-\rho)\frac{\lambda}{2}\|\beta\|_2^2 \tag{1}$$

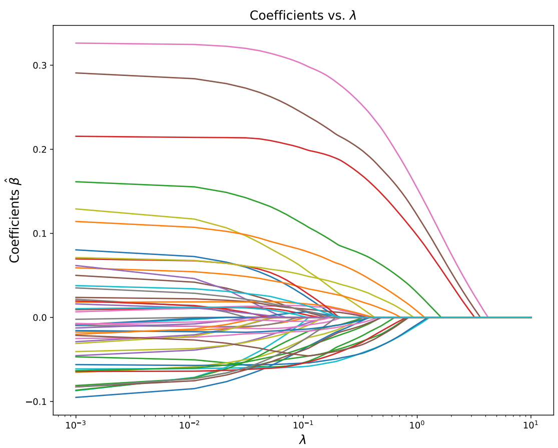
where $0 \leq \rho \leq 1$ controls how much emphasis is put on the ℓ_1 norm $\|\beta\|_1 = \sum_{j=1}^p \|\beta_j|$ relative to the ℓ_2 norm $\|\beta\|_2 = \sqrt{\sum_{j=1}^p \beta_j^2}$; for $\rho = 1$ we get the lasso, for $\rho = 0$ we get ridge regression, and for $0 < \rho < 1$ we use a combination of the two penalties.

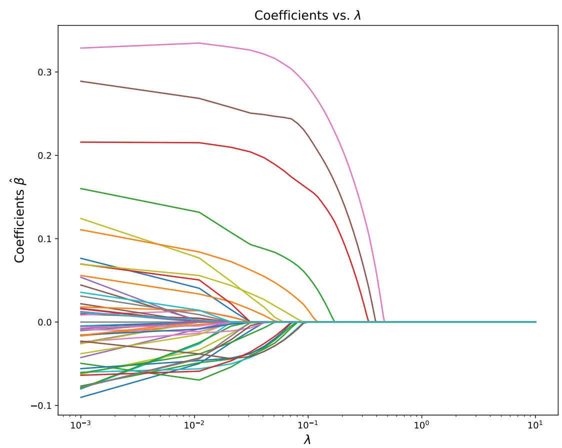
(a) Consider the special case

$$\widehat{\beta} = \operatorname*{argmin}_{\beta} \frac{1}{2}(Y - \beta)^2 + \rho\lambda|\beta| + (1-\rho)\frac{\lambda}{2}\beta^2$$

where Y is a single (random) number, and β is a single parameter. Give a closed-form solution for $\widehat{\beta}$.

(b) The plots below show the regularization paths for the elastic net with two different values ρ, specifically $\rho = 0.1$ and $\rho = 0.9$. Which is which? Why? Describe how the different regularizations affect the parameter estimates. (Note: the predictor variables are standardized.)





(c) Give an algorithm for solving the general elastic net optimization (1). Provide as much detail as you can.

3. ***How do I love CNNs? Let me count the ways*** (10 points)

Consider the following code for constructing a convolutional neural network:

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models

model = models.Sequential()
model.add(layers.Conv2D(100, (3, 3), input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((5, 5)))
model.add(layers.Flatten())
model.add(layers.Dense(1))
```

(a) The inputs are 32×32 color images, corresponding to tensors of shape $(32, 32, 3)$ with three color channels (R,G,B). For each of the layers, give a tuple that is the shape of the output tensor for that layer:

- `Conv2D`:
- `MaxPooling2D`:
- `Flatten`:
- `Dense`:

(b) For each of the layers, calculate the number of trainable parameters in that layer:

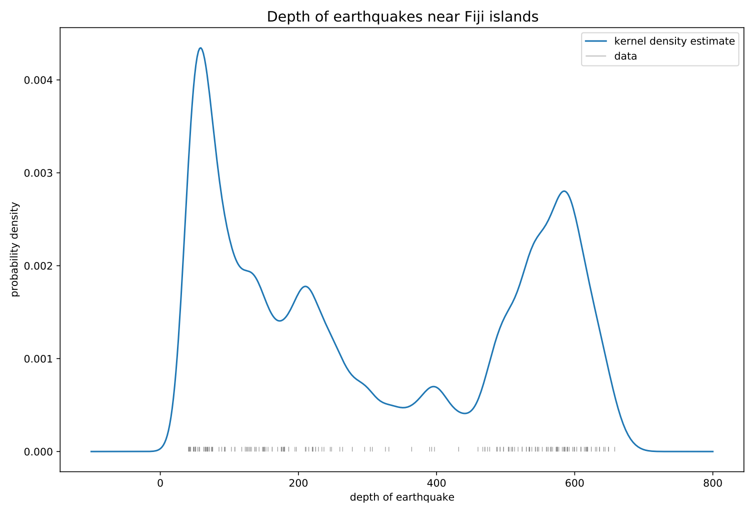
- `Conv2D`:
- `MaxPooling2D`:
- `Flatten`:
- `Dense`:

(c) What is missing from the above specification of the network?

4. ***Generative models: Give me an example*** (10 points)

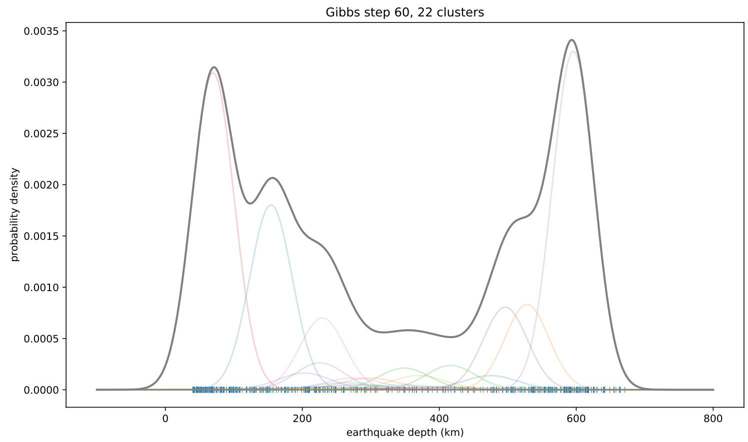
Density estimation gives a *generative model*, meaning that we can generate new data points from the distribution by sampling. This problem asks you to explain how to generate new samples for two density estimators.

(a) *Kernel density estimation.* Kernel density estimation is a frequentist method. On the Fiji earthquake data, the density might look like this:



Give an algorithm for generating a new sample X (a single random data point) from a kernel density estimate $\widehat{f}(x)$.

(b) *Dirichlet process mixture.* This is a Bayesian version of the kernel density estimator. On the Fiji earthquake data, the density might look like the following for a clustering generated by the Gibbs sampler:



Give an algorithm for generating a new sample X (a single random data point) from the density associated with a clustering obtained from the Gibbs sampler.

5. ***Implementation: Déjà vu all over again*** (10 points)

Consider the following partial code for Gaussian processes:

```
import numpy as np
from numpy.linalg import cholesky
import matplotlib.pyplot as plt

def gaussian_sample(mu, Sigma):
    A = cholesky(Sigma)
    Z = np.random.normal(loc=0, scale=1, size=len(mu))
    return np.dot(A, Z) + mu

def mean(n):
    return np.zeros(n)

def kernel(x, z, h=1):
    K = np.zeros(len(x)*len(z)).reshape(len(x), len(z))
    for j in np.arange(K.shape[1]):
        K[:,j] = (1/h)*np.exp(-(x-z[j])**2/(2*h**2))
    return K

# plot posterior sample, posterior mean, and 95% confidence

def sample_posterior(X, y, sigma2):
    xs = np.linspace(-5, 5, 500)
    K = kernel(X, X)
    Ks = kernel(X, xs)

    # your code begins
    ...
    posterior_mean = ...
    posterior_covariance = ...
    # your code ends

    var = np.diag(posterior_covariance)
    fs = gaussian_sample(posterior_mean, posterior_covariance)

    # the rest is plotting
    plt.figure(figsize=(10,7))
    plt.fill_between(xs, posterior_mean - 2*np.sqrt(var),
                         posterior_mean + 2*np.sqrt(var), alpha=.2)
    plt.plot(xs, posterior_mean, linewidth=3)
    plt.plot(xs, fs, color='gray', linewidth=1)
    plt.scatter(X, y, linewidth=2)
```

(a) Complete the code by implementing the computation of the posterior mean and the posterior covariance. Write your code below.

(b) Sketch what the plots would look like for the following two calls to `sample_posterior`:

```
sample_posterior(X=np.array([0]), y=np.array([0]), sigma2=1e-6)
sample_posterior(X=np.array([0]), y=np.array([0]), sigma2=1e-1)
```