EE-UY 4423: Introduction to Machine Learning
Midterm 1, Fall 2016

1. (a) a is bpm and b is in bpm/minute.

   (b) We use the regression formula:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i = \frac{1}{5}[0+0+1+2+3]$$

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i = \frac{1}{5}[75+65+90+110+130]$$

$$s_{xx} = \frac{1}{N}\sum_{i=1}^{N}(x_i-\bar{x})^2, \quad s_{yy} = \frac{1}{N}\sum_{i=1}^{N}(y_i-\bar{y})^2,$$

$$s_{xy} = \frac{1}{N}\sum_{i=1}^{N}(x_i-\bar{x})(y_i-\bar{y}).$$

   Then,

$$b = \frac{s_{xy}}{s_{xx}}, \quad a = \bar{y} - b\bar{x}.$$

   (c) Take the x-intercept at $a \approx 70$ and a slope of

$$b \approx (130-70)/(3-0) = 20.$$

   (d) The predicted heart rate would be

$$\hat{y} = 70 + (20)(30) = 670 \text{ bpm}.$$

   This is obviously unreasonable. The linear model is clearly not valid for large x.

   (e) You can create the scatter plot with the regression line with the following code.

```
xp = np.array([0,3])   # points for the regression line
yp = a + b*xp
plt.plot(x,y,'o')
plt.plot(xp,yp,'-')
```

2. (a) The first three rows would be:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0.01 & 0.1 \\ 1 & 120 & 1 & 0.8 \\ 1 & 92 & 3 & 0.6 \\ \vdots & \vdots & \vdots & \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 100 \\ 400 \\ 700 \\ \vdots \end{bmatrix}.$$

(b) The point corresponds to $\mathbf{x} = (0, 1, 0.5)$ so the predicted value would be

$$\hat{y} = 50 + 2(0) + 25(1) + 300(0.5) = 375 \text{ mW}.$$

(c) The variance is

$$\begin{aligned}
\mathrm{var}(\hat{y}) &= \frac{\sigma_\epsilon^2}{N} \phi(\mathbf{x})^\mathsf{T} R_{\phi\phi}^{-1} \phi(\mathbf{x}) \\
&= \frac{100^2}{100} [1, 0, 1, 0.5] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 10^{-4} & 0 & 0 \\ 0 & 0 & 0.1 & 0.02 \\ 0 & 0 & 0.02 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0.5 \end{bmatrix} \\
&= (100) \left[1 + 0(10)^{-4} + 0.1(1)^2 + 2(0.02)(1)(0.5) + 2(0.5)^2\right].
\end{aligned}$$

Since there is no under-modeling, the MSE is

$$\mathrm{MSE}(\hat{y}) = \sigma_\epsilon^2 + \mathrm{var}(\hat{y}) = 100^2 + \mathrm{var}(\hat{y}),$$

where $\mathrm{var}(\hat{y})$ is given above.

(d) Replace x_3 with two new variables: x_{3a} and x_{3b} with the following encoding:

Data service	x_{3a}	x_{3b}
WiFi	Data rate	0
Cellular	0	Data rate

Then, define the new feature vector, $\phi(\mathbf{x}) = [1, x_1, x_2, x_{3a}, x_{3b}]^\mathsf{T}$. The first three rows of the new feature matrix $\mathbf{A}$ will be

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0.01 & 0.1 & 0 \\ 1 & 120 & 1 & 0.8 & 0 \\ 1 & 92 & 3 & 0 & 0.6 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

3. (a) Take $\boldsymbol{\beta} = [a, b]^\mathsf{T}$ and $\phi(x, k) = [1, x\rho^k]^\mathsf{T}$.

(b) The feature matrix will have 15 rows given by

$$\mathbf{A} = \begin{bmatrix} 1 & (10)(0.9)^0 \\ 1 & (10)(0.9)^1 \\ \vdots & \vdots \\ 1 & (10)(0.9)^4 \\ 1 & (20)(0.9)^0 \\ 1 & (20)(0.9)^1 \\ \vdots & \vdots \\ 1 & (20)(0.9)^9 \end{bmatrix}$$

(c) There will be no under-modeling with $\rho = \rho_0$.

(d) The vector $\mathbf{f}_0$ is given by

$$\mathbf{f}_0 = \begin{bmatrix} a_0 + b_0(10)(\rho_0)^0 \\ a_0 + b_0(10)(\rho_0)^1 \\ \vdots \\ a_0 + b_0(10)(\rho_0)^4 \\ a_0 + b_0(20)(\rho_0)^0 \\ a_0 + b_0(20)(\rho_0)^1 \\ \vdots \\ a_0 + b_0(20)(\rho_0)^9 \end{bmatrix}.$$

Assuming $\mathbf{f}_0$ and $\mathbf{A}$ have been constructed and stored in `numpy` arrays `f0` and `A`, we can compute $\bar{z}_k$ from the code:

```
# Test point
x = 30
k = 2

# phi at test point
phi = np.array([1,x*(rho**k)])

# Compute expected value of the prediction
N = A.shape[0]
Rpp  = (1/N)*A.T.dot(A)
Rpf0 = (1/N)*A.T.dot(f0)
betabar = np.linalg.solve(Rpp,Rpf0)
zbar = phi.dot(betabar)
```

(e) For any fixed ρ, we saw above that we can find a and b from least-squares. So a simple solution is to discretize values of ρ, and for each value ρ find a least-squares fit. Then, select the value that minimizes the RSS. Here is some python code that can do the fit.

```
def fitmodel(x,k,z):
    """
    Finds the optimal rho, a and b for a model,

       zhat = a + b*(rho**k)
    """
    rhos = np.linspace(0,1,100)
    RSSs = []
    betas = []
    for rho in rhos:
        # A matrix
        phi1 = np.ones(15)
        phi2 = x* (rho**k)
        A = np.column_stack((phi1,phi2))

        # Find least squares fit
```

```
        betahat = np.linalg.lstsq(A,z)[0]

        # Compute prediction and RSS
        zhat = A.dot(betahat)
        RSS = np.mean( (z-zhat)**2)
        RSSs.append(RSS)
        betas.append(betahat)

    # Find the minimum RSS and corresponding value in rho
    RSSs = np.array(RSSs)
    imin = np.argmin(RSSs)
    rho = rhos[imin]
    betahat = betas[imin]
    a = betahat[0]
    b = betahat[1]
    return a, b, rho
```