EE-UY/CS-UY 4563: Introduction to Machine Learning
Midterm 1, Fall 2017

1. (34 points total)

   (a) (8 points) We can use the model, $f(x) = \beta_1 x + \beta_2 x^2$. We do not have a constant term to insure that $f(0) = 0$.

   (b) (9 points) For $x \geq 100$, we want $f(x) =$ constant. But, since $f(x)$ is continuous, we must have $f(x) = f(100)$ for $x \geq 100$. Thus, we can use the function,

$$f(x) = \begin{cases} \beta_1 x + \beta_2 x^2 & \text{if } x \leq 100, \\ \beta_1(100) + \beta_2(100)^2 & \text{if } x > 100. \end{cases} \tag{1}$$

   In this way, the function is quadratic for $x \leq 100$ and it takes the value $f(100)$ for all $x \geq 100$. If you give the answer (1), you will get full credit. But, there are multiple ways you can write the function. For example, you can write the function in (1) as

$$f(x) = \beta_1 z + \beta_2 z^2, \quad z = \min(x, 100). \tag{2}$$

   The only requirement to get full credit is that the function satisfies the required condition and has only two parameters.

   (c) (8 points) Using the parametrization in (2), the matrix $\mathbf{A}$ should be

$$\mathbf{A} = \begin{bmatrix} z_1 & z_1^2 \\ z_2 & z_2^2 \\ z_3 & z_3^2 \\ \vdots & \vdots \end{bmatrix} = \begin{bmatrix} 20 & 20^2 \\ 70 & 70^2 \\ 100 & (100)^2 \\ \vdots & \vdots \end{bmatrix},$$

   The response vector is the difference of heart rates,

$$\mathbf{y} = \begin{bmatrix} 80 - 70 \\ 90 - 75 \\ 85 - 55 \\ \vdots \end{bmatrix} = \begin{bmatrix} 10 \\ 25 \\ 30 \\ \vdots \end{bmatrix}$$

   (d) (9 points) One simple code is:

```
ntr = 50  # number of training samples
z = np.minimum(dosage[:ntr], 100)
Atr = np.column_stack((z, z**2))
ytr = hr_before[:ntr] - hr_after[:ntr]
```

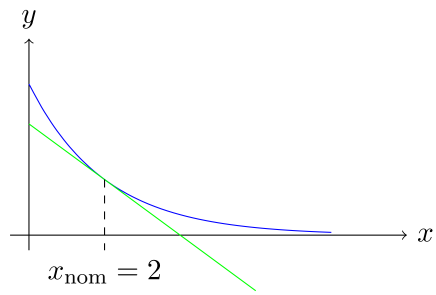

Figure 1: True function (blue) and estimated linear function (green).

2. (33 points total)

   (a) (6 points) Since the true data is exponential, it is not in the model class which is linear.

   (b) (9 points) The linear approximation is

   $$y_i = f_0(x_i) \approx f_0(x_{\text{nom}}) + f_0'(x_{\text{nom}})(x_i - x_{\text{nom}}).$$

   This fits the linear model with

   $$\begin{aligned} \beta_1 &= f'(x_{\text{nom}}) = -CDe^{-Dx_{\text{nom}}} \\ \beta_0 &= f_0(x_{\text{nom}}) - x_{\text{nom}} f'(x_{\text{nom}}) = C(1 - Dx_{\text{nom}})e^{-Dx_{\text{nom}}}. \end{aligned}$$

   Now, when the true model is linear and there is no noise, linear estimation will always obtain the ‘true parameters. Hence, we will obtain $\hat{\boldsymbol{\beta}} = (\beta_0, \beta_1)$ as above.

   (c) (9 points) See Fig. 1. Note that the true and estimated functions match exactly at $x_{\text{nom}} = 2$. The bias grows as x moves away from x_{nom}, so the largest bias would occur at $x = 4$.

   (d) (9 points) One possible solution is as follows:

```
# Split into training and test
ntr = 50
xtr = x[:ntr]
ytr = y[:ntr]
xts = x[ntr:]
yts = y[ntr:]

# Loop over different model orders
dtest = range(1,6)
rss = []
for d in dtest:
    bethat = fit(xtr,ytr,d)         # Fit the model on the training data
    yhat = predict(xts,betahat)     # Predict on test data
    rssi = np.sum((yts - yhat)**2) # Measure the test RSS
    rss.append(rssi)
```

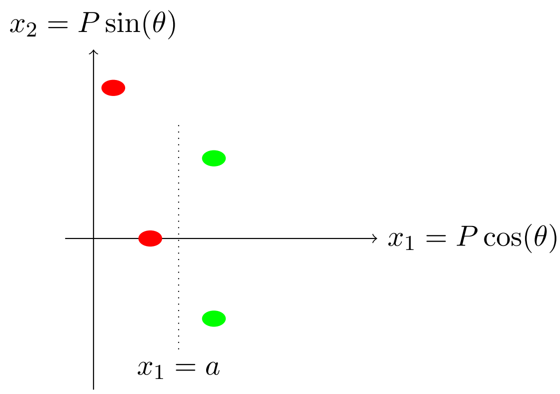

Figure 2: Scatter plot of the data points where the green circles are $y_i = 1$ (detected) and red circles are $y_i = 0$ (not detected). The dotted line is the boundary of a potential linear classifier.

```
# Find the minimum test RSS
iopt = np.argmin(rss)
dopt = dtest[iopt]
```

3. (33 points)

   (a) (8 points) See Fig. 2. The points are most easily drawn by recognizing that P is the radius and θ is the angle from the x_1-axis. For example, the two green points are at ± 45 degrees with radius 3.

   (b) (8 points) You can see that they are easily linearly separable. For example, we can use a linear classifier on a vertical line,

$$\hat{y} = \begin{cases} 1 & \text{if } x_1 \geq a, \\ 0 & \text{if } x_1 < a, \end{cases}$$

   for a suitable threshold level a. To select a, we need that $x_1 \geq a$ for the two green circles,

$$a \leq P\cos(\theta) = 3\cos(\pm 45) = \frac{3}{\sqrt{2}}.$$

   We also need that a is to the right of the red circle at $(P, \theta) = (1, 0)$ so

$$a \geq 1.$$

   So, the linear classifier will separate the classes as long as $a \in (1, 3/\sqrt{2})$. If you selected any single value of a that works, or any other line that works, you will get full credit.

   (c) (8 points) The z score is given by

$$z = \beta_0 + \beta_1 P\cos(\theta) + \beta_2 P\sin(\theta).$$

   To maximize $P(y = 1|\mathbf{x})$, we want to maximize z. Take the derivative with respect to θ:

$$z' = -\beta_1 P\sin(\theta) + \beta_2 P\cos(\theta) = 0 \Rightarrow \theta = \tan^{-1}(\beta_2/\beta_1).$$

You will get full credit for this solution. But, note that there is another point where $z' = 0$: $\theta = \tan^{-1}(\beta_2/\beta_1) + \pi$. You can verify that this point will give the smallest z, (i.e. the minimum probability of detection). But, you do not need to discuss this.

(d) (9 points) One solution is:

```
def predict(P,theta,beta):
    x1 = P*np.cos(theta)
    x2 = P*np.sin(theta)
    z = beta[0] + beta[1]*x1 + beta[2]*x2
    py = 1/(1+np.exp(-z))
    return py
```