EL-GY 9123: Introduction to Machine Learning
Midterm, Fall 2017

Answer all FOUR questions. Exam is closed book. No electronic aids. But, you are permitted two double-sided cheat sheets (four sides total). Part marks are given. If you do not remember a particular python command or its syntax, use pseudo-code and state what syntax you are assuming.

Best of luck!

1. An engineer wishes to calibrate a light sensor. To this end, she decides to model the sensor output voltage y in Volts (V) as a function of:

   - u = intensity of the light on the sensor in μW/cm^2; and
   - λ = wavelength of the light in nanometers (nm).

   She collects data as shown in Table 1. Note that only the first three samples are shown.

Trial ID	Voltage y (V)	Intensity u (μW/cm^2)	Wavelength λ (nm)
1	0.3	40	600
2	0.7	80	700
3	1.3	140	900
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 1: Training data for the model. Only the first three data records are shown.

   (a) The engineer first tries to model y using the light intensity u only, ignoring λ. Write a model $\hat{y} = f(u)$ that is linear in the light intensity u with the property that $\hat{y} = 0$ when $u = 0$. State what are the parameters in the model.

   (b) Next, she considers a more complex model $\hat{y} = f(u, \lambda)$ where

$$f(u, \lambda) = A + B(\lambda)u, \quad B(\lambda) = \begin{cases} B_1 + B_2\lambda & \text{if } \lambda < 800 \text{ nm}, \\ B_3 & \text{if } \lambda \geq 800 \text{ nm}, \end{cases}$$

   for parameters A, B_1, B_2 and B_3. She still wishes that $\hat{y} = f(u, \lambda) = 0$ when $u = 0$ for all λ. She also wants to force that $f(u, \lambda)$ is continuous. In particular, there is no discontinuity at $\lambda = 800$ nm. Describe the set of possible functions $f(u, \lambda)$ as a linear model,

$$f(u, \lambda) = \sum_{j=1}^{p} \beta_j \phi_j(u, \lambda),$$

   for basis functions $\phi_j(u, \lambda)$ and parameters β_j.

   (c) Using the model from part (b), find a matrix $\mathbf{A}$ such that $\hat{\mathbf{y}} = \mathbf{A}\boldsymbol{\beta}$, where $\hat{\mathbf{y}}$ is a vector of predicted values, $\boldsymbol{\beta}$ is the parameter vector. Given the data in Table 1, write the first three rows of the matrix $\mathbf{A}$.

   (d) Suppose you are given python vectors `u`, `lam`, `ytrue` and `beta` representing u, λ, the true values of y and vector of parameters $\boldsymbol{\beta}$. Write a python function `evaluate` that computes the predicted values $\hat{y}$ using the model in part (b) and returns the mean squared error:

$$\text{MSE} := \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2.$$

2. Suppose we consider two models from predicting a scalar variable y from a scalar input x:

$$\text{Model 1:} \quad \hat{y} = ae^{-bx},$$
$$\text{Model 2:} \quad \hat{y} = ae^{-bx} + c,$$

for parameters $\boldsymbol{\beta} = (a, b)$ for Model 1 and $\boldsymbol{\beta} = (a, b, c)$ for Model 2.

(a) Suppose the true data is of the form

$$y = f_0(x) = D\left[1 - e^{-Gx}\right], \tag{1}$$

for some $D > 0$ and $G > 0$. Is $f_0(x)$ in the model class 1 or 2? If it is in either model class, state the true parameters a, b and c in terms of D and G.

(b) Suppose we are given training data (x_i, y_i), $i = 1, \ldots, n$, where x_i is uniformly spaced in that

$$x_i = i\Delta, \tag{2}$$

for some step-size Δ. Also, suppose that the output follows Model 1 exactly in that,

$$y_i = a_0 e^{-b_0 x_i}, \tag{3}$$

for some true parameters a_0, b_0. Find a constant C such that

$$\hat{b} = C \ln\left[\frac{1}{N-1}\sum_{i=1}^{N-1} \frac{y_{i+1}}{y_i}\right], \tag{4}$$

provides an unbiased estimate of b. That is, $\hat{b} = b_0$. The parameter C may depend on Δ.

(c) Now suppose that (x_i, y_i) follows Model 2 exactly in that,

$$y_i = a_0 e^{-b_0 x_i} + c_0, \tag{5}$$

for some true parameters $\boldsymbol{\beta}_0 = (a_0, b_0, c_0)$ with $b_0 > 0$. The training data x_i is still uniformly spaced as in (2). Suppose we use the estimate for c,

$$\hat{c} = y_{i_0}, \tag{6}$$

for some *fixed* i_0. What is the bias $\hat{c} - c_0$ as a function of i_0, Δ and the true parameters $\boldsymbol{\beta}_0$. How should you select i_0 to minimize the bias?

(d) You are given functions

```
betahat = fit(x,y,model)
yhat = predict(x,beta,model)
```

where `fit` finds parameters `betahat` using training data `x,y` for `model=1` or `2`. The function `predict` computes the predicted values `yhat` based on the data `x`, parameters `beta` and `model=1` or `2`. Using these two functions, write python code that

- Given data `x` and `y`, splits the data into 50 training samples and the remaining samples being for test.
- Fits models for Models 1 and 2.
- Selects the Model 1 or 2 with lowest test RSS.

Note: You do not need to implement the one SE rule. Just pick the lowest test RSS. Also, you do not need to do K-fold validation. Just use validation with one training and test split.

3. A researcher wishes to build a classifier to predict a binary label $y = 0, 1$ from variables $u = (u_1, u_2)$. To fit the model, she collects data (u_{i1}, u_{i2}, y_i), $i = 1, \ldots, n$. The first four data points are shown below.

i	u_{i1}	u_{i2}	y_i
1	0.5	6	0
2	2	2	1
3	3	0.5	0
4	4	0.75	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Based on the data, she considers a classifier on the transformed variables,

$$(x_{i1}, x_{i2}) = (u_{i1}, u_{i1}u_{i2}).$$

(a) Draw a scatter plot of the transformed points (x_{i1}, x_{i2}) using a different marker for the two values of y.

(b) Are the first four data points linearly separable? If so, provide at least one classifier that is linear in $\mathbf{x}$ that separates the two classes. If not, find a classifier that makes the least number of errors on the first four points.

(c) Consider a logistic model for the data,

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + e^{-z}}, \quad z = \beta_0 + \beta_1 x_1 + \beta_2 x_2,$$

with coefficients $\boldsymbol{\beta} = (-3, 1, 2)$. Mathematically describe the set of points $\mathbf{u} = (u_1, u_2)$ such that $P(y = 1|\mathbf{u}) > 0.8$. Approximately draw this region in the (u_1, u_2) plane in the region $u_1 > 0$ and $u_2 > 0$.

You do not need to work out any logarithms. Just draw the rough shape of the boundary of the region and indicate the side of the boundary on which $P(y = 1|\mathbf{u}) > 0.8$.

(d) Consider a linear classifier

$$\hat{y} = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{if } z < 0, \end{cases} \qquad z = \beta_0 + \beta_1 x_1 + \beta_2 x_2.$$

Suppose we are given training data (x_{i1}, x_{i2}), $i = 1, \ldots, n$ and coefficients β_1, β_2 and we just need to determine the intercept β_0. Write a python program that finds the smallest value β_0 such that the missed detection rate, P_{MD}, on the training data is less 0.1. Recall that the missed detection rate is

$$P_{MD} := \frac{|\{i|\hat{y}_i = 0, y_i = 1\}|}{|\{i|y_i = 1\}|},$$

which is the fraction of samples where $y_i = 1$ and $\hat{y}_i = 0$ out of all the samples where $y_i = 1$. *Note:* Full credit requires that your solution is not of complexity $O(n^2)$. But, if that is all you can come up with, you will get most of the credit.

4. Consider a polynomial model of the form,

$$\hat{y} = \sum_{j=0}^{d} w_j x^j,$$

for parameters $\mathbf{w} = (w_0, \ldots, w_d)$. To fit this model with training data (x_i, y_i), $i = 1, \ldots, n$, we consider two possible loss functions:

$$J_{\text{ls}}(\mathbf{w}) := \sum_{i=1}^{n} (\hat{y}_i - y_i)^2, \quad J_{\text{log}}(\mathbf{w}) := \sum_{i=1}^{n} (\ln(\hat{y}_i) - \ln(y_i))^2.$$

(a) Find a matrix $\mathbf{A}$ and functions $g_i(z_i)$ such that the log least-squares objective can be written as

$$J_{\text{log}}(\mathbf{w}) = \sum_{i=1} g_i(z_i), \quad \mathbf{z} = \mathbf{A}\mathbf{w}.$$

(b) Describe how you would compute the gradient $\nabla J_{\text{log}}(\mathbf{w})$?

(c) For an initial condition, suppose we must choose between two possible parameter values: $\mathbf{w} = \mathbf{w}^A$ or $\mathbf{w}^B$. Suppose there are two samples and the predicted values $\hat{y}_i$ for both parameters are as follows: Which parameter, $\mathbf{w}^A$ or $\mathbf{w}^B$, would be selected under the

True y_i	$\hat{y}_i$ for $\mathbf{w} = \mathbf{w}^A$	$\hat{y}_i$ for $\mathbf{w} = \mathbf{w}^B$
1	2	8
100	200	100

least-squares loss $J_{\text{ls}}(\mathbf{w})$? Which parameter would be selected under the log least-squares loss $J_{\text{log}}(\mathbf{w})$? Explain.

Hint: In computing the log least-squares loss, it may be convenient to switch the base of the logarithm to make the calculations easy since you don't have a calculator.

(d) A classic way to speed up gradient descent is the *momentum* gradient descent method:

$$\mathbf{g}^{k+1} = \beta \mathbf{g}^k + \nabla J(\mathbf{w}^k)$$
$$\mathbf{w}^{k+1} = \mathbf{w}^k - \alpha \mathbf{g}^{k+1},$$

defined for parameters $\alpha > 0$ and $\beta \in [0, 1)$ and an initial condition $\mathbf{g}^0 = 0$. Complete the following function to implement the momentum gradient descent for an *arbitrary* loss function $J(\mathbf{w})$ – not necessarily one of the loss functions above.

```python
def momentum_grad(feval,...):
    ...
```

You will need to describe all the inputs to your function. You should include a function argument `feval` that returns the loss function and gradient (i.e. `J, Jgrad = feval(w)`). Make any other assumptions as necessary. But, write clear comments to explain the arguments that others can understand them. The function should return the final estimate $\mathbf{w}^k$ and loss function $J(\mathbf{w}^k)$.