ECE-GY 6143: Introduction to Machine Learning
Final Exam Solutions, Fall 2020

Prof. Sundeep Rangan

- This exam is done at home.
- You may use the Internet, your computer, class notes, labs, etc.
- You may *not* speak to others for help.
- Scan your solutions, convert to PDF and submit on Gradescope.
- Leave time for the submission.

Best of luck!

1. *SVM.* An SVM classifier is trained on 4×4 images, $\mathbf{X}_i$ with binary labels $y_i = \pm 1$. Three of the images in the training data set are:

$$\mathbf{X}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{X}_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \mathbf{X}_3 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

with labels $y_1 = 1, y_2 = 1, y_3 = -1$. You are given a test image,

$$\mathbf{X} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

(a) The training and test images, $\mathbf{X}_i$ and $\mathbf{X}$, are converted to 16-dimensional vectors $\mathbf{x}_i$ and $\mathbf{x}$. Find the squared distances

$$d_i = \|\mathbf{x} - \mathbf{x}_i\|^2, \quad i = 1, 2, 3.$$

(b) Suppose a kernel is

$$K(\mathbf{x}, \mathbf{x}_i) = \max\{0, 1 - \|\mathbf{x}_i - \mathbf{x}\|^2 \gamma\}.$$

What value of $\gamma > 0$ guarantees that $K(\mathbf{x}, \mathbf{x}_i) > 0$ for $y_i = 1$ and $K(\mathbf{x}, \mathbf{x}_i) = 0$ for $y_i = -1$ for $i = 1, 2, 3$.

(c) Complete the following python function,

```
def predict(Xtr,ytr,X,...):
    """
    Parameters
    Xtr:  Training data, shape (ntr,nrow,ncol)
    ytr:  Training labels, shape (ntr,)
    X:    Test data, shape (n,nrow,ncol)
    ...   Add other parameters as needed

    Returns:
    yhat: Predicted labels, shape (n,)
    """
    ...
    return yhat
```

to perform the SVM classification,

$$z = \sum_{i=1}^{n} \alpha_i y_i K(x, x_i) + b, \quad \widehat{y} = \begin{cases} 1, & \text{if } z > 0 \\ -1, & \text{otherwise.} \end{cases}$$

For full credit, use python broadcasting.

2. *Neural Networks.* Consider a neural network, with $N_i = 2$ input units, $\mathbf{x} = (x_1, x_2)$, $N_h = 3$ hidden units with ReLU activations and one output unit for regression,

$$z_j^{\mathrm{H}} = \sum_{k=1}^{N_i} W_{jk}^{\mathrm{H}} x_k + b_j^{\mathrm{H}}, \quad u_j^{\mathrm{H}} = \max\{0, z_j^{\mathrm{H}}\}, \quad j = 1, \ldots, N_h$$
$$\widehat{y} = \sum_{k=1}^{N_h} W_k^{\mathrm{O}} u_k^{\mathrm{H}} + b^{\mathrm{O}},$$

(a) Suppose that

$$\mathbf{W}^{\mathrm{H}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & -1 \end{bmatrix}, \quad \mathbf{b}^{\mathrm{H}} = \begin{bmatrix} -0.5 \\ -1 \\ 3 \end{bmatrix}$$

Write equations for z_j^{H} in terms of x for $j = 1, 2, 3$.

(b) Draw the region of inputs (x_1, x_2) where $u_j^{\mathrm{H}} > 0$ for all j.

(c) You are given data `X`, `y` as well as weights and biases `Wh,bh` for the hidden layer. Write a few lines of python code to fit `Wo,bo` for the output layer by minimizing the MSE. You may assume you have a function

```
beta = lstsq(A,b) #  Solves the least squares problem A.dot(beta) = b
```

3. *Backpropagation.* Consider the model with an M-dimensional input $\mathbf{x} = (x_1, \ldots, x_M)$ and scalar output $\widehat{y}$ given by

$$z_k = \sum_{j=1}^{M} A_{kj} x_j + b_k, \quad u_k = \frac{e^{z_k}}{\sum_{\ell=1}^{K} e^{z_\ell}}, \quad k = 1, \ldots, K$$
$$\widehat{y} = \sum_{k=1}^{K} \sum_{j=1}^{M} C_{kj} x_j u_k + \sum_{k=1}^{K} d_k u_k$$

with parameters A_{kj}, b_k, C_{kj} and d_k.

(a) You are given training samples $\mathbf{x}_i = (x_{i1}, \ldots, x_{iM})$ and y_i. Write the equations relating the inputs x_{ij} to the output $\widehat{y}_i$.

(b) Draw the computation graph between the data, parameters and loss function using the loss function,

$$J = \sum_{i=1}^{N} (y_i - \widehat{y}_i)^2.$$

(c) Show how to compute the components of $\partial J / \partial z$ from the components of $\partial J / \partial u$.

4. *CNN dimensions.* A neural network for image processing has the following input and first two layers:

   - `Input`: 128×256 RGB image
   - `Conv1`: 2D convolution, $K = 9 \times 9$ filters, $N_o = 64$ output channels, same size output, ReLU activation
   - `MaxPool1`: 2D max pooling, $K = 4 \times 4$ pool size, horizontal and vertical stride $s = 2$.

   (a) What are the dimensions of the input and the outputs for the first two layers for a mini-batch of 32 images.

   (b) How many parameters are there in the `Conv1` and `MaxPool1` layers?

   (c) If the value of only the red channel on a single pixel of an input sample is changed, what is the maximum number of output values that could be changed in the outputs of the `Conv1` and `MaxPool1` layers?

5. *Transfer learning.* You want to build a classifier that detects bicycles in images. In each sample, the input to the classifier is a set of 5 photos of some scene taken at different angles. Each photo is an RGB image of size 128× 128. The output of the classifier is a single binary value, 0 or 1, for each sample, indicating if there was a bicycle in *any* of the five photos in the sample.

   (a) You want to write the classifier that takes a set of samples `X` and return a set of predictions `yhat`. What should the shape of `X` and `yhat` be?

   (b) You are given an excellent pre-trained image classifier that works on a single 128× 128 RGB image. The pre-trained classifier was trained on 1000 image classes and can be accessed by:

```
Z = pretrained.predict(X)  # Outputs the logits for the 1000 classes
```

   Suppose that in the pre-trained classifier, bicycles were one of the image classes, say class 50. Write a possible implementation of a classifier using the pretrained model:

```
def classify(X,pretrained):
    ...
    return yhat
```

   There is no single correct answer. Do something reasonable with an explanation.

   (c) Now suppose that bicycles were *not* one of the image classes that the pretrained classifier uses. So, you decide to transfer learn using the outputs of one of the layers near the end of the pre-trained model using simple logistic regression on those outputs. Assume you have the following functions:

```
# Outputs the 2000 hidden units at some layer near the
# end of the pre-trained model
Z = pretrain_base.predict(X)

reg = LogisticRegression()  # Builds a logistic regression object
reg.fit(X,y)   # Fits the logistic model from X to y
yhat = reg.predict(X)  # Predicts binary outputs
```

   Write functions to fit and predict the pre-trained model:

```
def fit(Xtr,ytr,...)
    ...
    return ...

def predict(X, ...)
    ...
    return yhat
```

   The functions should take whatever inputs and outputs are necessary.

6. *PCA.* Let $\mathbf{X}$ is a data matrix of $n = 100$ samples, each of dimension $p = 200$ with one sample per row. Let $\tilde{\mathbf{X}}$ be the matrix with the mean μ removed from the each row. Suppose it has a truncated SVD of the form,

$$\frac{1}{\sqrt{n}}\tilde{X} \approx \mathbf{U}\text{diag}(\mathbf{s})\mathbf{V}^{\mathsf{T}},$$

where

$$\mathbf{U} = \begin{bmatrix} 0.5 & 0.3 & 0.1 \\ 0.6 & 0.4 & 0.2 \\ \vdots & \vdots & \vdots \end{bmatrix}, \quad \mathbf{s} = [10, 8, 2],$$

The columns of $\mathbf{U}$ and $\mathbf{V}$ are orthonormal. Only the first two rows of $\mathbf{U}$ are shown. Let $\mathbf{v}_j$, $j = 1, \ldots, 3$ be the three columns of $\mathbf{V}$.

(a) What are the dimensions of $\mathbf{U}$ and $\mathbf{V}$?

(b) Suppose $\mathbf{x}_1$ is the first row of $\mathbf{X}$. Write $\mathbf{x}_1$ as a linear combination of the mean μ and the three PCs $\mathbf{v}_j$, $j = 1, 2, 3$.

(c) Suppose that $\widehat{\mathbf{x}}_1$ is the approximation of $\mathbf{x}_1$ constructed using only μ, and the first PC $\mathbf{v}_1$. What is $\|\widehat{\mathbf{x}}_1 - \mathbf{x}_1\|^2$?

7. *K-means.* You are given training data $(\mathbf{x}_i, y_i)$ with continuous scalar-valued outputs y_i. To train the model, you decide to:

   - Perform K-means on the training data $\mathbf{x}_i$ resulting in cluster centers $\boldsymbol{\mu}_k$, $k = 1, \ldots, K$.
   - Compute $\overline{y}_k$, the mean of the y_i's in each cluster k.

   Then, given a new input $\mathbf{x}$ you make a prediction:

$$\widehat{y} = \frac{\sum_{k=1}^{K} \overline{y}_k e^{-\gamma \|\mathbf{x}-\boldsymbol{\mu}_k\|^2}}{\sum_{k=1}^{K} e^{-\gamma \|\mathbf{x}-\boldsymbol{\mu}_k\|^2}}$$

   The number of clusters K and the scaling value $\gamma > 0$ are fixed (they are not trained). In the questions below, you can assume you have the following functions:

```
km = KMeans(n_cluster=nc)   # Creates a K-Means object
km.fit(X)  # Fits the k-means clusters
km.predict(X)   # Returns the index of the closest cluster for each row in X
km.cluster_centers_   # Returns the cluster centers (one cluster per row)
```

   (a) Suppose a test sample $\mathbf{x}$ is close to samples in a cluster of four points with outputs $y_i = 4, 4, 5, 6$. The sample $\mathbf{x}$ is far from the other cluster centers. Approximately, what is $\widehat{y}$? For full marks, provide an explanation.

   (b) Write a function to perform the training:

```
def fit(Xtr,ytr,...)
    ...
    return ...
```

   The function should take whatever arguments are necessary and return the relevant parameters in the model.

   (c) Write a function to compute a vector of values `yhat` on new data `X`.

```
def predict(X,...)
    ...
    return yhat
```

   Add whatever input arguments are needed.