Data Mining and Machine Learning
Fall 2018, Homework 2
(due on Sep 11, 11.59pm EST)

Jean Honorio `jhonorio@purdue.edu`

The homework is based on a total of 10 points. Your code **should be in Python 2.7**. For clarity, the algorithms presented here will assume zero-based indices for arrays, vectors, matrices, etc. Please read the submission instructions at the end. **Failure to comply to the submission instructions will cause your grade to be reduced.**

In this homework, we will focus on classification for separable data. You can use the following script **createsepdata.py** to create some synthetic separable data:


```
import numpy as np
import scipy.linalg as la
# Input: number of samples n
#        number of features d
# Output: numpy matrix X of features, with n rows (samples), d columns (features)
#             X[i,j] is the j-th feature of the i-th sample
#         numpy vector y of labels, with n rows (samples), 1 column
#             y[i] is the label (+1 or -1) of the i-th sample
# Example on how to call the script:
#     import createsepdata
#     X, y = createsepdata.run(10,3)
def run(n,d):
  y = np.ones((n,1))
  y[n/2:] = -1
  X = np.random.random((n,d))
  idx_row, idx_col = np.where(y==1)
  X[idx_row,0] = 0.1+X[idx_row,0]
  idx_row, idx_col = np.where(y==-1)
  X[idx_row,0] = -0.1-X[idx_row,0]
  U = la.orth(np.random.random((d,d)))
  X = np.dot(X,U)
  return (X,y)
```

In this homework, you will implement algorithms that depend on a kernel function K. You should call the following script **K.py**

```
import numpy as np
import math
# Input: numpy vector x of d rows, 1 column
#        numpy vector z of d rows, 1 column
# Output: kernel K(x,z) = exp(-1/2 * norm(x-z)^2)
# Example on how to call the script:
#     import K
#     v = K.run( np.array([[1], [4], [3]]) , np.array([[2], [7], [-1]]) )
def run(x,z):
  x = x.flatten()
  z = z.flatten()
  if x.size != z.size:
    raise ValueError
  return math.exp(-1/2 * np.sum((x-z) ** 2))
```


Here are the questions:

1) [4 points] Implement the following perceptron algorithm with kernels, introduced in Lecture 3.

Input: number of iterations L, training data $x_t \in \mathbb{R}^d$, $y_t \in \{+1, -1\}$ for $t = 0, \ldots, n-1$
Output: $\alpha \in \mathbb{R}^n$

$\alpha \leftarrow 0$
for iter $= 1, \ldots, L$ **do**
  for $t = 0, \ldots, n-1$ **do**
    if $y_t \left(\sum_{i=0}^{n-1} \alpha_i y_i K(x_i, x_t) \right) \leq 0$ **then**
      $\alpha_t \leftarrow \alpha_t + 1$
    end if
  end for
end for

The header of your **Python script kerperceptron.py** should be:

```
# Input: number of iterations L
#        numpy matrix X of features, with n rows (samples), d columns (features)
#            X[i,j] is the j-th feature of the i-th sample
#        numpy vector y of labels, with n rows (samples), 1 column
#            y[i] is the label (+1 or -1) of the i-th sample
# Output: numpy vector alpha of n rows, 1 column
def run(L,X,y):
  # Your code goes here
  return alpha
```

2) [2 points] Implement the following predictor function with kernels, introduced in Lecture 3.

Input: $\alpha \in \mathbb{R}^n$, training data $x_t \in \mathbb{R}^d$, $y_t \in \{+1, -1\}$ for $t = 0, \ldots, n-1$, testing point $z \in \mathbb{R}^d$
Output: label $\in \{+1, -1\}$
if $\sum_{i=0}^{n-1} \alpha_i y_i K(x_i, z) > 0$ **then**
  label $\leftarrow +1$
else
  label $\leftarrow -1$
end if

The header of your **Python script kerpred.py** should be:

```
# Input: numpy vector alpha of n rows, 1 column
#        numpy matrix X of features, with n rows (samples), d columns (features)
#            X[i,j] is the j-th feature of the i-th sample
#        numpy vector y of labels, with n rows (samples), 1 column
#            y[i] is the label (+1 or -1) of the i-th sample
#        numpy vector z of d rows, 1 column
# Output: label (+1 or -1)
def run(alpha,X,y,z):
  # Your code goes here
  return label
```


3) [4 points] Now we ask you to implement the following dual support vector machines (DSVM) problem, introduced in Lecture 4.

$$\begin{aligned} &\text{maximize } \sum_{i=0}^{n-1} \alpha_i - \frac{1}{2} \sum_{i,j=0}^{n-1} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ &\text{subject to } \alpha_i \geq 0 \text{ for } i = 0, \ldots, n-1 \end{aligned}$$

Let $f = (-1, -1, \ldots, -1)^{\mathrm{T}} \in \mathbb{R}^n$ be an n-dimensional vector of minus ones. Let $b = (0, 0, \ldots, 0)^{\mathrm{T}} \in \mathbb{R}^n$ be an n-dimensional vector of zeros. Let $A \in \mathbb{R}^{n \times n}$ be a matrix with n rows and n columns, with minus ones on the diagonal ($a_{i,i} = -1$ for $i = 0, \ldots, n-1$) and zeros on the off-diagonal ($a_{i,j} = 0$ for $i \neq j$). In other words, A is the negative of the identity matrix. Let $H \in \mathbb{R}^{n \times n}$ be a matrix with n rows and n columns, where $h_{i,j} = y_i y_j K(x_i, x_j)$ for all $i, j = 0, \ldots, n-1$. Since $\alpha \in \mathbb{R}^n$, we can rewrite the DSVM problem as:

$$\begin{aligned} &\text{minimize } \frac{1}{2} \alpha^{\mathrm{T}} H \alpha + f^{\mathrm{T}} \alpha \\ &\text{subject to } A\alpha \leq b \end{aligned}$$

Fortunately, the package **cvxopt** can solve exactly the above problem by doing:

```
import cvxopt as co
alpha = np.array(co.solvers.qp(co.matrix(H,tc='d'),co.matrix(f,tc='d'),
                               co.matrix(A,tc='d'),co.matrix(b,tc='d'))['x'])
```

The header of your **Python script kerdualsvm.py** should be:

```
# Input: numpy matrix X of features, with n rows (samples), d columns (features)
#            X[i,j] is the j-th feature of the i-th sample
#        numpy vector y of labels, with n rows (samples), 1 column
#            y[i] is the label (+1 or -1) of the i-th sample
# Output: numpy vector alpha of n rows, 1 column
def run(X,y):
  # Your code goes here
  return alpha
```

Notice that for prediction you can reuse the **kerpred.py** script that you wrote for question 2.

SOME POSSIBLY USEFUL THINGS.
Python 2.7 is available at the servers antor and data. From the terminal, you can use your Career account to start a ssh session:


```
ssh username@data.cs.purdue.edu
OR
ssh username@antor.cs.purdue.edu
```


From the terminal, to start Python:

```
python
```

Inside Python, to check whether you have **Python 2.7**:

```
import sys
print (sys.version)
```

Inside Python, to check whether you have the package **cvxopt**:

```
import cvxopt
```

From the terminal, to install the Python package **cvxopt**:

```
pip install --user cvxopt
OR
python -m pip install --user cvxopt
```

More information at `https://cvxopt.org/install/index.html`

SUBMISSION INSTRUCTIONS.
Your code **should be in Python 2.7**. We **only need** the Python scripts (.py files). We **do not need** the Python (compiled) bytecodes (.pyc files). You will get 0 points if your code does not run. You will get 0 points in you fail to include the Python scripts (.py files) even if you mistakingly include the bytecodes (.pyc files). We will deduct points, if you do not use the right name for the Python scripts (.py) as described on each question, or if the input/output matrices/vectors/scalars have a different type/size from what is described on each question. Homeworks are to be solved individually. We will run plagiarism detection software.
Please, submit a single ZIP file **through Blackboard**. Your Python scripts (**kerperceptron.py**, **kerpred.py**, **kerdualsvm.py**) should be directly inside the ZIP file. **There should not be any folder inside the ZIP file**, just Python scripts. The ZIP file should be named according to your Career account. For instance, if my Career account is jhonorio, the ZIP file should be named **jhonorio.zip**