In [1]:
```
"""
Landon Buell
Marek Petrik
CS 750.01
21 March 2020
"""

import numpy as np
```

Problem 1 [33%]

In this problem, we will establish some basic properties of vectors and linear functions.

1. The L2 norm of a vector measures the length (size) of a vector. The norm for a vector x of size n is defined as:

$$||x||_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

Show that the norm can be expressed as the following quadratic expression:

$$||x||_2^2 = x^T x$$

Given the definition of the L_2 norm above, we can compute the square of it. THis removes the radical, and then the norm, squared is just the sum of all elements in the vector x, squared. Thus:

$$||x||_2^2 = \sum_{i=1}^{n} x_i^2$$

We can compare this to the vector x, by itself, it is a column vector with n entres, shaped $1 \times n$. The corresponding transpose, x^T, then also has n elements, but is a row vector containing the same elements, shaped $m \times 1$. When computing their matrix multiplication, or in this case, dot product, $x^T x$. The result is the i-th element in each array multiplied by the same element in the other array, and them summed. Given that they are just mutual transposes, corresponding elemtns are identitcial so:

$$x_i^T = x_i \rightarrow x_i^T x_i = x_i^2$$

Finally, summing over all elements in the arrays just produces the result:

$$||x||_2^2 = \sum_{i=1}^{n} x_i^2$$

Which we have shown above to be equivalent to the square of the L_2 norm.

2. Let a and x be vectors of size $n = 3$ and consider the following linear function $f(x) = a^T x$. Show that the gradient of f is: $\nabla_x f(x) = a$

When evaluating $f(x) = a^T x$, we see that it becomes the dot product of vectors a and x:

$$a^T x = a_1 x_1 + a_2 x_2 + a_3 x_3$$

So, if we take the gradient of $f(x)$, with respect to each elment in the x vector:

$$\nabla_x f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \end{bmatrix}$$

Using rules of partial differntiation from calculus 3, we can then evaluate this gradient to be:

$$\nabla_x f(x) = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Which is indentically the vector a

3. Let A be a symmetric matrix of size 3×3 and consider the following quadratic function $f(x) = x^T Ax$. Show that the gradient of f is: $\nabla_x f(x) = 2Ax$. A matrix is symmetric if $A_{ij} = A_{ji}$ for all i and j.

Evaluatiing out $A\vec{x}$ is:

$$Ax = \begin{bmatrix} A_{11} A_{12} A_{13} \\ A_{21} A_{22} A_{23} \\ A_{31} A_{32} A_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + A_{13}x_3, \\ A_{21}x_1 + A_{22}x_2 + A_{23}x_3, \\ A_{31}x_1 + A_{32}x_2 + A_{33}x_3 \end{bmatrix}$$

Which we can then left multiply by x^T:

$$x^T(Ax) = \Big(x_1(A_{11}x_1 + A_{12}x_2 + A_{13}x_3)\Big) + \Big(x_2(A_{21}x_1 + A_{22}x_2 + A_{13}x_3)\Big) + \Big(x_3(A_{31}x_1 +$$

Which also:

$$x^T(Ax) = \Big(A_{11}x_1^2 + A_{12}x_2x_1 + A_{13}x_3x_1\Big) + \Big(A_{21}x_1x_2 + A_{22}x_2^2 + A_{23}x_3x_2\Big) + \Big(A_{31}x_1x_3 +$$

We can exploit the symmetry of the matrix A, and replace all elements below the main diagonal with the corresponding elements above the main diagonal:

$$x^T(Ax) = \Big(A_{11}x_1^2 + A_{12}x_2x_1 + A_{13}x_3x_1\Big) + \Big(A_{12}x_1x_2 + A_{22}x_2^2 + A_{23}x_3x_2\Big) + \Big(A_{13}x_1x_3 +$$

Combine like-terms:

$$f(x) = x^T(Ax) = \Big(A_{11}x_1^2 + 2A_{12}x_2x_1 + 2A_{13}x_3x_1\Big) + \Big(A_{22}x_2^2 + 2A_{23}x_3x_2\Big) + \Big(A_{33}x_3^2\Big)$$

We can then take the gradient of $f(x)$ as defined above. This, along with the chain rule of calculus:

$$\nabla_x f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \end{bmatrix} = \begin{bmatrix} 2A_{11}x_1 + 2A_{12}x_2 + 2A_{13}x_3 \\ 2A_{12}x_1 + 2A_{22}x_2 + 2A_{23}x_3 \\ 2A_{13}x_1 + 2A_{23}x_2 + 2A_{33}x_3 \end{bmatrix}$$

Once again, exploit the symmetry of the matrix to restore incidies, and then remove the common factor of 2 from all matrix elements, and then decpmosing the system back into a matrix-vector equation:

$$\nabla_x \left[f(x) \right] = 2 \begin{bmatrix} A_{11} A_{12} A_{13} \\ A_{21} A_{22} A_{23} \\ A_{31} A_{32} A_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Which is mathematically equaivalent to $2Ax$, thus $\nabla_x \left[f(x) \right] = \nabla_x \left[x^T Ax \right] = 2Ax$

Problem 2 [34%]

Hint: You can follow the slides from the March 4th class, or the LAR reference from the class website. See the class website for some recommended linear algebra references.

You will derive the formula used to compute the solution to ridge regression. The objective in ridge regression is:

$$f(\beta) = ||y - A\beta||_2^2 + \lambda||\beta||_2^2$$

Here, β is the vector of coefficients that we want to optimize, A is the design matrix, y is the target, and $\lambda$is the regularization coefficient. The notation $|| \cdot ||_2$ represents the Euclidean (or L_2) norm. Our goal is to find β that solves:

$$\min_\beta f(\beta)$$

1. Express the ridge regression objective $f(\beta)$ in terms of linear and quadratic terms. Recall that $||\beta||_2^2 = \beta^T \beta$.The result should be similar to the objective function of linear regression.

Expand $f(\beta)$ as a polynomial:

$$f(\beta) = \left(y - A\beta \right)^T \left(y - A\beta \right) + \lambda \left(\beta^T \beta \right)$$

Then 'distribute' out the transposition:

$$f(\beta) = \left(y^T - \beta^T A^T \right) \left(y - A\beta \right) + \lambda \left(\beta^T \beta \right)$$

Evaluate ('foil') out to build the polynomial:

$$f(\beta) = y^T y - y^T A\beta - \beta^T A^T y + \beta^T A^T A\beta + \lambda \left(\beta^T \beta \right)$$

And then simplify, using rules of transposition:

$$f(\beta) = y^T y - 2y^T A\beta + \beta^T A^T A\beta + \lambda \left(\beta^T \beta \right)$$

2.Derive the gradient: $\nabla_\beta \left[f(\beta) \right]$ using the linear and quadratic terms above.

The value of $\nabla_\beta \left[f(\beta) \right]$ is:

$$\left[\frac{df}{d\beta} \right] = \left[-2y^T A + \beta^T A^T A + \lambda\beta^T \right]$$

3. Since f is convex, its minimal value is attained when :

$$\nabla_\beta \left[f(\beta) \right] = 0$$

Derive the expression for β that satisifies the inequality above.

We can transpose the gradient of the function f, and equate it to 0^T, which is still 0:

$$-2A^T y + 2A^T A\beta + \lambda\beta = 0$$

Thus $f(\beta) = 0$ when:

$$\beta = \frac{2A^T y}{2A^T A + \lambda}$$

4. Implement the algorithm for computing β and use it on a small dataset of your choice. Do not forget about the intercept.

let $\lambda = 1$. Let A be the feature matrix

In [ ]:

```

```

Problem 3 [33%]

Using the MNIST dataset, which we used already in Assignment 2, compare whether boosting, bagging, and random forests work the best. You may may want to use only a subset of the data. Optional: Use xgboost (also available for Python) to see whether the results are better than other boosting methods.

In [11]:

```python
# Load in MNIST data
from sklearn.datasets import fetch_openml
from sklearn.metrics import classification_report

X,y = fetch_openml('mnist_784',version=1,return_X_y=True)
X_train,y_train = X[:10000],y[:10000]
X_test,y_test = X[10000:15000],y[10000:15000]
print(X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
```

```
(10000, 784) (10000,)
(5000, 784) (5000,)
```

In [12]:
```
""" Sklearn Boosting """

from sklearn.ensemble import GradientBoostingClassifier

GB_CLF = GradientBoostingClassifier()
GB_CLF.fit(X_train,y_train)

y_pred = GB_CLF.predict(X_test)

report = classification_report(y_test,y_pred,labels=[0,1,2,3,4,5,6,7,8,9])
print(report)
```

```
              precision    recall  f1-score   support

           0       0.97      0.96      0.97       495
           1       0.95      0.98      0.97       563
           2       0.88      0.90      0.89       471
           3       0.92      0.86      0.89       516
           4       0.90      0.93      0.91       488
           5       0.91      0.91      0.91       455
           6       0.93      0.93      0.93       476
           7       0.94      0.94      0.94       523
           8       0.88      0.88      0.88       488
           9       0.90      0.89      0.90       525

   micro avg       0.92      0.92      0.92      5000
   macro avg       0.92      0.92      0.92      5000
weighted avg       0.92      0.92      0.92      5000


C:\Users\Landon\Anaconda3\lib\site-packages\numpy\lib\arraysetops.py:565: F
utureWarning: elementwise comparison failed; returning scalar instead, but
in the future will perform elementwise comparison
  mask &= (ar1 != a)
```

In [13]:
```
""" Sklearn Bagging """

from sklearn.ensemble import BaggingClassifier

Bag_CLF = BaggingClassifier()
Bag_CLF.fit(X_train,y_train)

y_pred = Bag_CLF.predict(X_test)

report = classification_report(y_test,y_pred,labels=[0,1,2,3,4,5,6,7,8,9])
print(report)
```

```
              precision    recall  f1-score   support

           0       0.92      0.96      0.94       495
           1       0.95      0.98      0.97       563
           2       0.84      0.87      0.85       471
           3       0.85      0.84      0.84       516
           4       0.89      0.90      0.89       488
           5       0.89      0.86      0.88       455
           6       0.93      0.90      0.91       476
           7       0.94      0.92      0.93       523
           8       0.84      0.83      0.84       488
           9       0.87      0.84      0.86       525

   micro avg       0.89      0.89      0.89      5000
   macro avg       0.89      0.89      0.89      5000
weighted avg       0.89      0.89      0.89      5000


C:\Users\Landon\Anaconda3\lib\site-packages\numpy\lib\arraysetops.py:565: F
utureWarning: elementwise comparison failed; returning scalar instead, but
in the future will perform elementwise comparison
  mask &= (ar1 != a)
```

In [14]:
```
""" Sklearn Random Forests """

from sklearn.ensemble import RandomForestClassifier

RF_CLF = RandomForestClassifier()
RF_CLF.fit(X_train,y_train)

y_pred = RF_CLF.predict(X_test)

report = classification_report(y_test,y_pred,labels=[0,1,2,3,4,5,6,7,8,9])
print(report)
```

```
C:\Users\Landon\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:245:
FutureWarning: The default value of n_estimators will change from 10 in ver
sion 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)

              precision    recall  f1-score   support

           0       0.94      0.98      0.96       495
           1       0.94      0.98      0.96       563
           2       0.84      0.90      0.87       471
           3       0.83      0.85      0.84       516
           4       0.87      0.90      0.88       488
           5       0.91      0.87      0.89       455
           6       0.95      0.91      0.93       476
           7       0.93      0.93      0.93       523
           8       0.88      0.80      0.84       488
           9       0.89      0.85      0.87       525

   micro avg       0.90      0.90      0.90      5000
   macro avg       0.90      0.90      0.90      5000
weighted avg       0.90      0.90      0.90      5000


C:\Users\Landon\Anaconda3\lib\site-packages\numpy\lib\arraysetops.py:565: F
utureWarning: elementwise comparison failed; returning scalar instead, but
in the future will perform elementwise comparison
  mask &= (ar1 != a)
```

In running the three models form the sklearn.ensembele sub-module, I took the precision, recall and F1 scores for each method. The results after running each model a few different times shows that the Gradient Boosting Classification method seems to produce the best results. Each time, it had the highest average score for eahc class between the three metrics. Unfortunately, this method also took the logest to fit and predict a data set.

Optional Problem 4 [+10%]

Hint: We did not cover this material in class, but it is important regardless. The slides from March 9th may be helpful when implementing this. In this problem, you will implement a gradient

descent algorithm for solving a linear regression problem. Recall that the RSS objective in linear regression is:

$$f(\beta) = ||y - A\beta||_2^2$$

1. Consider the problem of predicting revenue as a function of spending on TV and Radio advertising. There are only 4 data points:

$$\begin{bmatrix} \text{Revenue} & \text{TV} & \text{Radio} \\ 20 & 3 & 7 \\ 14 & 4 & 6 \\ 32 & 6 & 1 \\ 5 & 1 & 1 \end{bmatrix}$$

Write down the design matrix of predictors, A , and the response vector, y, for this regression problem. Do not forget about the intercept, which can be modeled as a predictor with a constant value over all data points. The matrix A should be 4×3 dimensional.

```
In [ ]:
```