Overview

1. Assignment 2 Question 3: Code Review
2. Matrix/Vector/Norm Notation
3. Minimizing Quadratic Functions as Linear Systems
4. Objective Function for the findMin.jl Function
5. Assignment 2 Question 4: Code Review
Assignment 2 Question 3: Code Review

```julia
include("kMeans.jl")
function quantizeImage(imageName,nBits)
    #load imageName
    I = imread(imageName) #Note that matrix I has three dimensions.

    #Form matrix X (by reshaping I) with pixels as rows and colours as columns.
    #Note that matrix X should have two dimensions.

    #Find the prototypes of matrix X.

    #Return the cluster assignments y, the means W, the number of rows in the image nRows,
    #and the number of columns nCols.
end

function deQuantizeImage(y,W, nRows, nCols)
    #Define a matrix Xhat with nRows * nCols rows and 3 columns.
    #Note that Xhat has two dimensions.

    #For all i, should assign the prototype of row i of X to row i of Xhat.

    #Reshape Xhat and return.
end
```
Express the following functions in terms of vectors, matrices, and norms (there should be no summations or maximums),

$$f(w) = \frac{1}{2} \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \frac{\lambda}{2} \sum_{j=1}^{d} w_j^2$$
Express the following functions in terms of vectors, matrices, and norms (there should be no summations or maximums),

\[ f(w) = \frac{1}{2} \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \frac{\lambda}{2} \sum_{j=1}^{d} w_j^2 \]

Recall, that all vectors are column-vectors,

\[ w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{id} \end{bmatrix} \]

- \( w_j \) is the scalar parameter \( j \).
- \( y_i \) is the label of example \( i \).
- \( x_i \) is the column-vector of features for example \( i \).
- \( x_{ij} \) is feature \( j \) in example \( i \).
Let's first focus on the **regularization term**, 

\[ f(w) = \frac{1}{2} \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \frac{\lambda}{2} \sum_{j=1}^{d} w_j^2 \]
Let's first focus on the regularization term,

\[ f(w) = \frac{1}{2} \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \frac{\lambda}{2} \sum_{j=1}^{d} w_j^2 \]

Recall the definition of inner product and L2-norm of vectors,

\[ \|v\| = \sqrt{\sum_{j=1}^{d} v_j^2}, \quad u^T v = \sum_{j=1}^{d} u_j v_j \]

Hence, we can write the regularizer in various forms using,

\[ \|w\|^2 = \sum_{j=1}^{d} w_j^2 = \sum_{j=1}^{d} w_j w_j = w^T w \]
Let’s next focus on the least squares term,

\[
f(w) = \frac{1}{2} \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \frac{\lambda}{2} \|w\|^2
\]

Let’s define the residual vector \( r \) with elements

\[
r_i = w^T x_i - y_i
\]

We can write the least squares term as squared L2-norm of residual,

\[
\sum_{i=1}^{n} (w^T x_i - y_i)^2 = \sum_{i=1}^{n} r_i^2 = r^T r = \|r\|^2
\]
Let’s next focus on the least squares term,

\[ f(w) = \frac{1}{2} \| r \|^2 + \frac{\lambda}{2} \| w \|^2, \quad \text{with } r_i = w^T x_i - y_i \]
Let's next focus on the least squares term,

\[ f(w) = \frac{1}{2} \| r \|^2 + \frac{\lambda}{2} \| w \|^2, \quad \text{with } r_i = w^T x_i - y_i \]

\[ X \text{ denotes the matrix containing the } x_i \text{ (transposed) in the rows:} \]

\[
X = \begin{bmatrix}
(x_1)^T \\
(x_2)^T \\
\vdots \\
(x_n)^T
\end{bmatrix}
\]
Let’s next focus on the least squares term, 

\[ f(w) = \frac{1}{2} \| r \|^2 + \frac{\lambda}{2} \| w \|^2, \quad \text{with} \quad r_i = w^T x_i - y_i \]

\( X \) denotes the matrix containing the \( x_i \) (transposed) in the rows:

\[
X = \begin{bmatrix}
(x_1)^T \\
(x_2)^T \\
\vdots \\
(x_n)^T
\end{bmatrix}
\]

Using \( w^T x_i = (x_i)^T w \) and the definitions of \( r, y, \) and \( X \):

\[
r = \begin{bmatrix}
r_1 \\
r_2 \\
\vdots \\
r_n
\end{bmatrix} = \begin{bmatrix}
w^T x_1 - y_1 \\
w^T x_2 - y_2 \\
\vdots \\
w^T x_n - y_n
\end{bmatrix} = \begin{bmatrix}
(x_1)^T w \\
(x_2)^T w \\
\vdots \\
(x_n)^T w
\end{bmatrix} - \begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_n
\end{bmatrix} = \begin{bmatrix}
(x_1)^T \\
(x_2)^T \\
\vdots \\
(x_n)^T
\end{bmatrix} w - y = Xw - y
Matrix/Vector/Norm Notation

- Let’s next focus on the least squares term,

\[ f(w) = \frac{1}{2} \| r \|_2^2 + \frac{\lambda}{2} \| w \|_2^2, \quad \text{with} \quad r_i = w^T x_i - y_i \]

- \( X \) denotes the matrix containing the \( x_i \) (transposed) in the rows:

\[
X = \begin{bmatrix}
(x_1)^T \\
(x_2)^T \\
\vdots \\
(x_n)^T
\end{bmatrix}
\]

- Using \( w^T x_i = (x_i)^T w \) and the definitions of \( r, y, \) and \( X \):

\[
r = \begin{bmatrix}
r_1 \\
r_2 \\
\vdots \\
r_n
\end{bmatrix} = \begin{bmatrix}
w^T x_1 - y_1 \\
w^T x_2 - y_2 \\
\vdots \\
w^T x_n - y_n
\end{bmatrix} = \begin{bmatrix}
(x_1)^T w \\
(x_2)^T w \\
\vdots \\
(x_n)^T w
\end{bmatrix} - \begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_n
\end{bmatrix} = \begin{bmatrix}
(x_1)^T \\
(x_2)^T \\
\vdots \\
(x_n)^T
\end{bmatrix} w - y = Xw - y
\]

- Therefore: \( f(w) = \frac{1}{2} \| Xw - y \|^2 + \frac{\lambda}{2} \| w \|^2. \)
A quadratic function is a function of the form

\[ f(w) = \frac{1}{2} w^T A w + b^T w + \gamma \]

for a square matrix \( A \), vector \( b \), and scalar \( \gamma \).
A quadratic function is a function of the form

\[ f(w) = \frac{1}{2} w^T A w + b^T w + \gamma \]

for a square matrix \( A \), vector \( b \), and scalar \( \gamma \).

Write the minimizer of the following function as a system of linear equations, using vector/matrix notation.

\[ f(w) = \frac{1}{2} \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \frac{\lambda}{2} \sum_{j=1}^{d} w_j^2 \]
A quadratic function is a function of the form

\[ f(w) = \frac{1}{2} w^T A w + b^T w + \gamma \]

for a square matrix \( A \), vector \( b \), and scalar \( \gamma \).

Write the minimizer of the following function as a system of linear equations, using vector/matrix notation.

\[ f(w) = \frac{1}{2} \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \frac{\lambda}{2} \sum_{j=1}^{d} w_j^2 \]

To minimize convex functions, it is sufficient to find \( w \) such that \( \nabla f(w) = 0 \).
Convert to vector/matrix form:

\[
\begin{align*}
    f(w) &= \frac{1}{2} \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \lambda \sum_{j=1}^{d} w_j^2 \\
    &= \frac{1}{2} (Xw - y)^T (Xw - y) + \frac{\lambda}{2} w^T w
\end{align*}
\]
Convert to vector/matrix form:

\[
f(w) = \frac{1}{2} \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \frac{\lambda}{2} \sum_{j=1}^{d} w_j^2 = \frac{1}{2} (Xw - y)^T (Xw - y) + \frac{\lambda}{2} w^T w
\]

\[
\rightarrow f(w) = \frac{1}{2} w^T X^T X w - w^T X^T y + \frac{1}{2} y^T y + \frac{\lambda}{2} w^T w
\]

Recall:

- For scalar value \( c \): \( \nabla_w [c] = 0 \) (column vector of zeros)
- For column vector \( b \): \( \nabla_w [w^T b] = b \)
- For symmetric matrix \( A \): \( \nabla_w [\frac{1}{2} w^T A w] = A w \)
Minimizing Quadratic Functions as Linear Systems

- Convert to vector/matrix form:

\[
f(w) = \frac{1}{2} \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \frac{\lambda}{2} \sum_{j=1}^{d} w_j^2 = \frac{1}{2} (Xw - y)^T (Xw - y) + \frac{\lambda}{2} w^T w
\]

\[
\rightarrow f(w) = \frac{1}{2} w^T X^T X w - w^T X^T y + \frac{1}{2} y^T y + \frac{\lambda}{2} w^T w
\]

- Recall:
  - For scalar value c: \( \nabla_w [c] = 0 \) (column vector of zeros)
  - For column vector b: \( \nabla_w [w^T b] = b \)
  - For symmetric matrix \( A \): \( \nabla_w [\frac{1}{2} w^T A w] = Aw \)

- Find \( w \) such that \( \nabla f(w) = 0 \):

\[
\nabla f(w) = X^T X w - X^T y + \lambda w = 0 \rightarrow (X^T X + \lambda I)w = X^T y
\]

- Note \( \nabla f(w) \) is a column vector with dimension \( d \times 1 \).
For the function below, in Julia, return the function value and its gradient with respect to $w$:

$$f(w) = \frac{1}{2} (Xw - y)^T (Xw - y) + \frac{\lambda}{2} w^T w$$
For the function below, in Julia, return the function value and its gradient with respect to $w$:

$$f(w) = \frac{1}{2} (Xw - y)^T (Xw - y) + \frac{\lambda}{2} w^T w$$

```julia
function ridgeRegressionObj(w,X,y)
    lambda = 1
    f = 1/2 * (X*w - y)'*(X*w - y) + lambda/2 * w'*w
    g = X'*X*w - X'*y + lambda*w
    return (f,g)
end
```
Assignment 2 Question 4: Code Review

```julia
# Load X and y variable
using JLD
data = load("basisData.jld")
(X,y,Xtest,ytest) = (data["X"],data["y"],data["Xtest"],data["ytest"])

# Fit a least squares model
include("leastSquares.jl")
model = leastSquares(X,y)

# Evaluate training error
yhat = model.predict(X)
trainError = mean((yhat - y).^2)
@printf("Squared train Error with least squares: %.3f\n", trainError)

# Evaluate test error
yhat = model.predict(Xtest)
testError = mean((yhat - ytest).^2)
@printf("Squared test Error with least squares: %.3f\n", testError)

# Plot model
using PyPlot
figure()
plot(X,y,"b."
Xhat = minimum(X):.:maximum(X)
yhat = model.predict(Xhat)
plot(Xhat,yhat,"g")
```

include("misc.jl")

```julia
function leastSquares(X,y)
    # Find regression weights minimizing squared error
    w = (X'X)\\(X'y)
    # Make linear prediction function
    predict(Xhat) = Xhat\\w
    # Return model
    return GenericModel(predict)
end
```