CPSC 340 Assignment 6 (due Friday November 29 at 11:55pm)

Answer:
Kexin Wen (76020940)
Yixuan LI (61261814)

1 Robust PCA

The function *example_RPCA* loads a dataset X where each row contains the pixels from a single frame of a video of a highway. The demo applies PCA to this dataset and then uses this to reconstruct the original image. It then shows the following 3 images for each frame of the first 50 frames (pausing for a tenth of a second on each frame):

1. The original frame.
2. The reconstruction based on PCA.
3. A binary image showing locations where the reconstruction error is non-trivial.

Recently, latent-factor models have been proposed as a strategy for "background subtraction": trying to separate objects from their background. In this case, the background is the highway and the objects are the cars on the highway. In this demo, we see that PCA does an ok job of identifying the cars on the highway in that it does tend to identify the locations of cars. However, the results aren't great as it identifies quite a few irrelevant parts of the image as objects.

Robust PCA is a variation on PCA where we replace the L2-norm with the L1-norm,

$$f(Z, W) = \sum_{i=1}^{n} \sum_{j=1}^{d} |w_j^T z_i - x_{ij}|,$$

and it has recently been proposed as a more effective model for background subtraction. Write a new function, *robustPCA*, that uses the Huber loss to approximate the absolute value to implement robust PCA. Hand in your code.

Hint: most of the work has been done for you in the function *PCA_gradient*. This function implements an alternating minimization approach to minimizing the PCA objective (without enforcing orthogonality). This gradient-based approach to PCA can be modified to use the Huber loss. A reasonable value of the hyper-parameter ϵ in the Huber loss might be 0.01 for this problem. You may want to use a smaller version of the dataset when debugging your objective/gradient code.

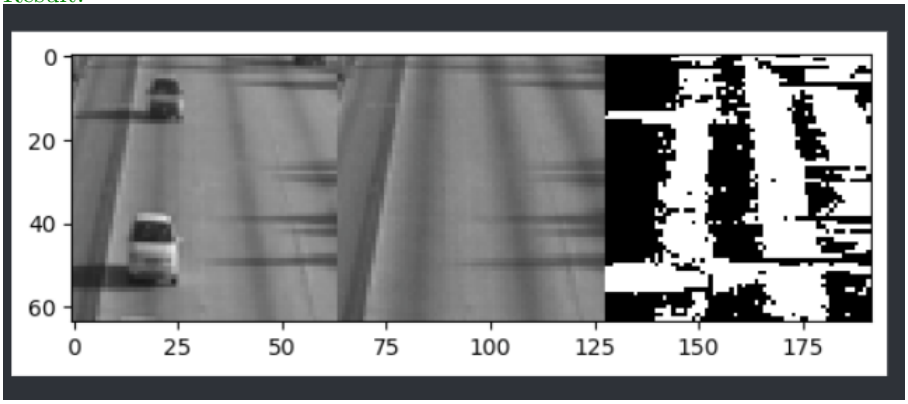
Answer:
Code:

```
function RobustPCA(X,k)
    (n,d) = size(X)
    # Subtract mean
    mu = mean(X,dims=1)
    X -= repeat(mu,n,1)
    W=randn(k,d)
    Z=randn(n,k)
    epsilon=0.01
    R=Z*W-X
    case=abs.(R).<=epsilon
    close=findall(case)
    far=findall(.!case)
    f=sum((1/2)R[close].^2)+epsilon*sum(abs.(R[far]).-(1/2)epsilon)
    funObjZ(z)=rpcaObjZ(z,X,W)
    funObjW(w)=rpcaObjW(w,X,Z)

    for iter in 1:50
        fOld = f
        Z[:]=findMin(funObjZ,Z[:],verbose=false,maxIter=10)
        W[:]=findMin(funObjW,W[:],verbose=false,maxIter=10)
        R=Z*W-X
        f=sum((1/2)R[close].^2)+epsilon*sum(abs.(R[far]).-(1/2)epsilon)
        @printf("Iteration %d, loss = %f\n",iter,f/length(X))
        if (fOld - f)/length(X) < 1e-2
            break
        end
    end

    compress(Xhat) = compress_gradientDescent(Xhat,W,mu)
    expand(Z) = expandFunc(Z,W,mu)

    return CompressModel(compress,expand,W)
end
```

Code:

```
function rpcaObjZ(z,X,W)
    # Rezie vector of parameters into matrix
    n = size(X,1)
    k = size(W,1)
    Z = reshape(z,n,k)
    epsilon=0.01
    R=Z*W-X
    case=abs.(R).<=epsilon
    close=findall(case)
    far=findall(.!case)
    R[far]=sign.(R[far])
    dR = R
    G = dR*W'
    f=sum((1/2)R[close].^2)+epsilon*sum(abs.(R[far]).-(1/2)epsilon)
    return (f,G[:])
end
function rpcaObjW(w,X,Z)
    # Rezie vector of parameters into matrix
    d = size(X,2)
    k = size(Z,2)
    W = reshape(w,k,d)
    # Comptue function value
    epsilon=0.01
    R=Z*W-X
    case=abs.(R).<=epsilon
    close=findall(case)
    far=findall(.!case)
    R[far]=sign.(R[far])
    dR = R
    G = Z'dR
    f=sum((1/2)R[close].^2)+epsilon*sum(abs.(R[far]).-(1/2)epsilon)
    # Return function and gradient vector
    return (f,G[:])
end
```

Result:



2 Multi-Dimensional Scaling

The function *example_MDS* loads the animals dataset and then applies gradient dsecent to minimize the following multi-dimensional scaling (MDS) objective (starting from the PCA solution):

$$f(Z) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=i+1}^{n}(\|z_i - z_j\| - \|x_i - x_j\|)^2. \tag{1}$$

The result of applying MDS is shown below.

Although this visualization isn't perfect (with "gorilla" being placed close to the dogs and "otter" being placed close to two types of bears), this visualization does organize the animals in a mostly-logical way.

2.1 ISOMAP

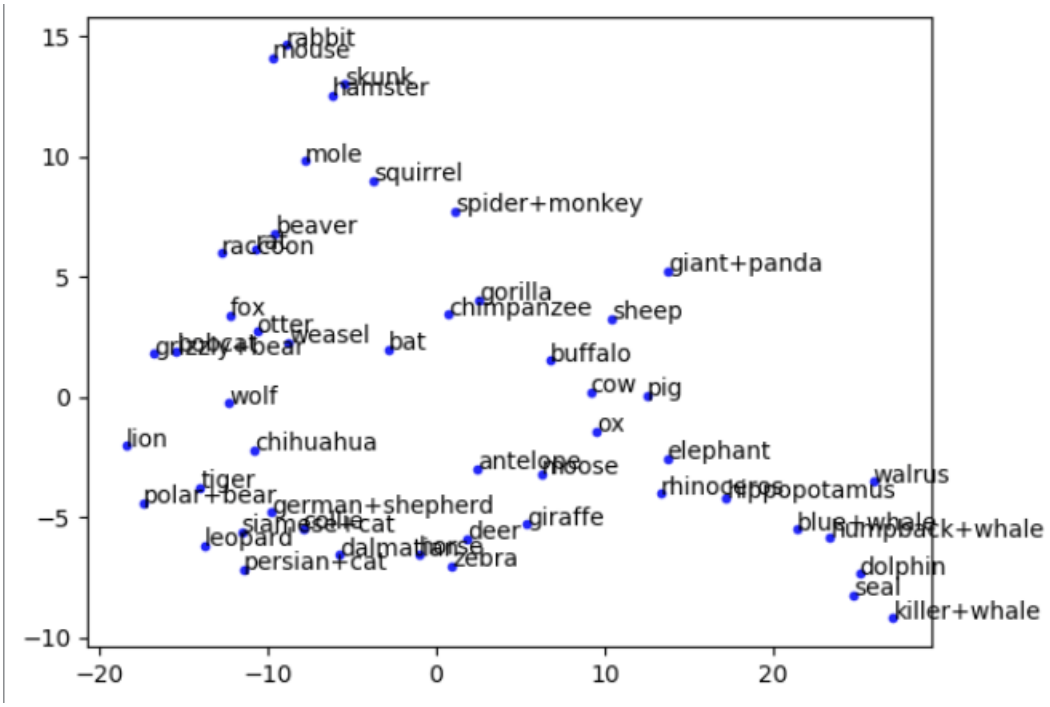
Euclidean distances between very different animals are unlikely to be particularly meaningful. However, since related animals tend to share similar traits we might expect the animals to live on a low-dimensional manifold. This suggests that ISOMAP may give a better visualization. Make a new function *ISOMAP* that computes the approximate geodesic distance (shortest path through a graph where the edges are only between nodes that are k-nearest neighbour) between each pair of points, and then fits a standard MDS model (1) using gradient descent. Hand in your code and the plot of the result when using the 3-nearest neighbours.

Hint: the function *dijskstra* (in *misc.jl*) can be used to compute the shortest (weighted) distance between two points in a weighted graph. This function requires an n by n matrix giving the weights on each edge (use ∞ as the weight for absent edges). Note that ISOMAP uses an undirected graph, while the k-nearest neighbour graph might be asymmetric. You can use the usual heuristics to turn this into an undirected graph of including an edge i to j if i is a KNN of j or if j is a KNN of i. (Also, be careful not to include the point itself in the KNN list).

Answer:
Code:

```
function ISOMAP(X)
    k =3
    (n,d) = size(X)
    # Compute all distances
    D = distancesSquared(X,X)
    D = sqrt.(abs.(D))
    E = ones(n,n).*Inf
    for i in 1:n
        A = sortperm(D[i,:])
        for j in 2:k+1
          E[i,A[j]] =D[i,A[j]]
          E[A[j],i] =D[A[j],i]
        end
    end
    D = zeros(n,n)
    for i in 1:n
        for j in 1:n
         D[i,j] = dijkstra(E,i,j)
        end
    end
    model = PCA(X,2)
    Z = model.compress(X)
    funObj(z) = stress(z,D)
    Z[:] = findMin(funObj,Z[:])
    return Z
end
```

Result:



2.2 ISOMAP with Disconnected Graph

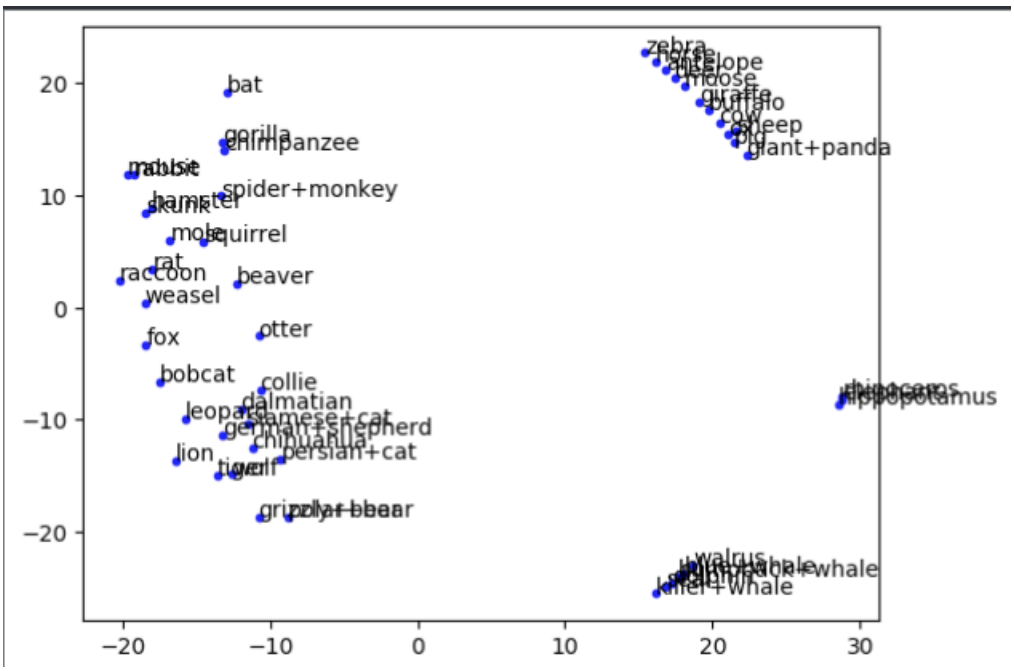
An issue with measuring distances on graphs is that the graph may not be connected. For example, if you run your ISOMAP code with 2-nearest neighbours then some of the distances are infinite. One heuristic to address this is to set these infinite distances to the maximum distance in the graph (i.e., the maximum geodesic distance between any two points that are connected), which will encourage non-connected points to be far apart. Modify your ISOMAP function to implement this heuristic. Hand in your code and the plot of the result when using the 2-nearest neighbours.

Answer:
Code:

```
function ISOMAP2(X)
    k =2
    (n,d) = size(X)
    # Compute all distances
    D = distancesSquared(X,X)
    D = sqrt.(abs.(D))
    E = ones(n,n).*Inf
    for i in 1:n
        A = sortperm(D[i,:])
        for j in 2:k+1
          E[i,A[j]] =D[i,A[j]]
          E[A[j],i] =D[A[j],i]
        end
    end
    D = zeros(n,n)
    max = -Inf
    for i in 1:n
        for j in 1:n
         D[i,j] = dijkstra(E,i,j)
         if(D[i,j]!=Inf && D[i,j] > max)
            max =  D[i,j]
         end
        end
    end
```

```
    D = zeros(n,n)
    max = -Inf
    for i in 1:n
        for j in 1:n
         D[i,j] = dijkstra(E,i,j)
         if(D[i,j]!=Inf && D[i,j] > max)
            max =  D[i,j]
         end
        end
    end
    for i in 1:n
        for j in 1:n
             if(D[i,j]==Inf )
             D[i,j] = max
         end
        end
    end
    # Initialize low-dimensional representation wi
    model = PCA(X,2)
    Z = model.compress(X)
    funObj(z) = stress(z,D)
    Z[:] = findMin(funObj,Z[:])
    return Z
end
```

Result:



3 Neural Networks

3.1 Neural Networks by Hand

Suppose that we train a neural network with sigmoid activations and one hidden layer and obtain the following parameters (assume that we don't use any bias variables):

$$W = \begin{bmatrix} -2 & 2 & -1 \\ 1 & -2 & 0 \end{bmatrix}, v = \begin{bmatrix} 3 \\ 1 \end{bmatrix}.$$

Assuming that we are doing regression, for a training example with features $x_i^T = \begin{bmatrix} -3 & -2 & 2 \end{bmatrix}$ what are the values in this network of z_i, $h(z_i)$, and $\hat{y}_i$?

Answer:
$z_i = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$
$h(z_i) = \begin{bmatrix} 0.5 \\ 0.731 \end{bmatrix}$
$\hat{y}_i = 1.5+0.731 = 2.231$

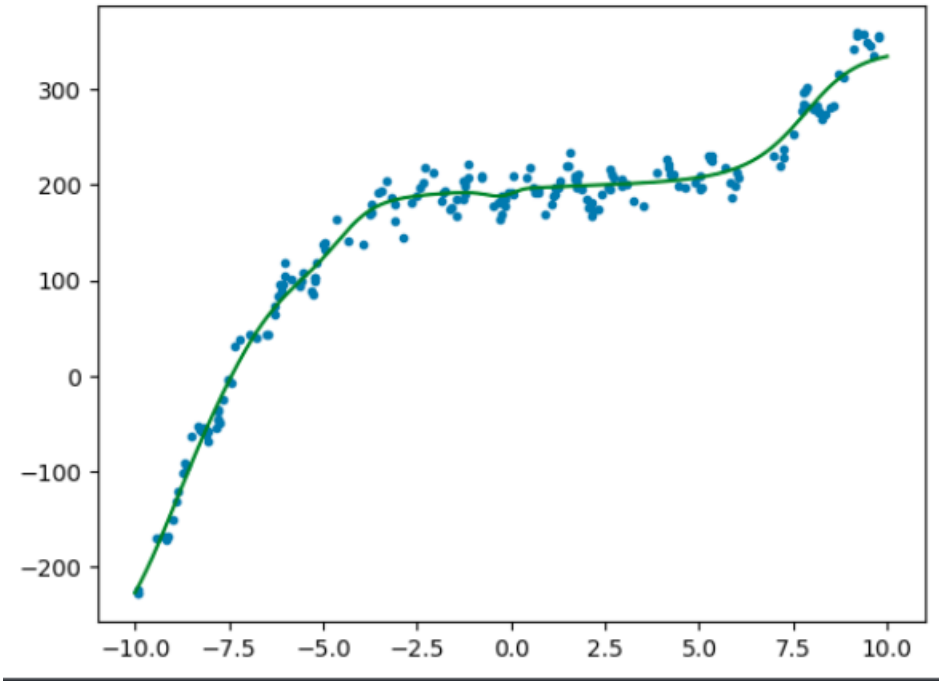
3.2 Neural Network Tuning - Regression

The file `example_nnet.jl` runs a stochastic gradient method to train a neural network on the *basisData* dataset from a previous assignment. However, in its current form it doesn't fit the data very well. Modify the training procedure to improve the performance of the neural network. Hand in your plot after changing the code to have better performance, and list the changes you made.

Hint: there are many possible strategies you could take to improve performance. Below are some suggestions, but note that the some will be more effective than others:

- Changing the network structure (*nHidden* is a vector giving the number of hidden units in each layer).
- Changing the training procedure (you can change the stochastic gradient step-size, use mini-batches, run it for more iterations, add momentum, switch to *findMin*, and so on).
- Transform the data by standardizing the feautres, standardizing the targets, and so on.
- Add regularization (L2-regularization, L1-regularization, dropout, and so on).
- Add bias variables within the hidden layers.
- Change the loss function or the non-linearities (right now it uses squared error and tanh to introduce non-linearity).
- Use mini-batches of data, possibly with batch normalization.

Answer:
Increasing $nHidden = 150$
Dynamic step size, going from $1e-4$ to $1e-7$
Set to 50000 iterations.



3.3 Neural Network Tuning - Classification

The file `example_usps.jl` runs a stochastic gradient method to train a neural network on a set of images of digits. Modify the training procedure to improve the performance of the neural network. List the changes you made and the best test performance that you were able to obtain.

Answer:
Result:

```
Training iteration = 58000, test error = 0.277000
```

List of Change:
Increase the number of hidden units to 100.
Manual babysitting of the step-size (Occasionally measure error, when the change of error is less than 0.01,

decrease step-size by $0.5 * 10^-4$).
Add momentum as (+0.9 * dalta w).

4 Very-Short Answer Questions

1. Is the NMF loss function convex? What is an optimization method you could use to try to minimize it?

   Answer: No. Projected gradient method.

2. Consider fitting a linear latent-factor model, using L1-regularization of the w_c values and L2-regularization of the z_i values,

$$f(Z,W) = \frac{1}{2}\|ZW - X\|_F^2 + \lambda_W \sum_{c=1}^{k} [\|w_c\|_1] + \frac{\lambda_Z}{2} \sum_{i=1}^{n} \left[\|z_i\|^2\right],$$

   (a) What is the effect of λ_Z on the two parts of the fundamental trade-off in machine learning?

   Answer: increasing λ_Z means stronger regularization and it will lead to lower approximation error but higher training error.

   (b) What is the effect of k on the two parts?

   Answer: increasing k means more complicated model, which leads to lower training error but higher approximation error.

   (c) Would either of answers to the *previous two questions* change if $\lambda_W = 0$?

   Answer: if $\lambda_W = 0$, λ_Z doesn't work, but k still has the effect.

3. Which is better for recommending movies to a new user, collaborative filtering or content-based filtering? Briefly justify your answer.

   Answer: Content based filtering, because this method extract features x_i of users and items in existing database, building model to predict rating y_i given x_i. The model can be applied to prediction for new user or new movie. New users havent rated anything, so collaborative filtering does not predict well for new user.

4. Is ISOMAP mainly used for supervised or unsupervised learning? Is it parametric or non-parametric?

   Answer: Supervised learning and nonparametric.

5. What is the difference between Euclidean distance and geodesic distance?

   Answer: When images are on a manifold in the high dimensional space. Geodesic distance is distance through space of rotations/ resizings. Euclidean distance doesnt reflect manifold structure.

6. Are neural networks mainly used for supervised or unsupervised learning? Are they parametric or nonparametric?

   Answer: Supervised learning and parametric.

7. The loss for a neural network is typically non-convex. Give one set of hyperparameters for which the loss is actually convex.

   Answer: If we don't use any activation funciton like sigmoid, then the neural network is actually a linear regression model, which has convex loss function.

8. Assuming we have some procedure that returns a random global minimum of a neural network objective, how does the depth of a neural network affect the fundamental trade-off? For a convolutional network, how would the width of the convolutions affect the fundamental trade-off?

   Answer: For neural network, with increasing depth, training error of global optima decreases, but training error may poorly approximate test error.
   For convolutional network, with increasing width. the model becomes complicated, training error is going to decrease but approximation error will increase.

9. What is the "vanishing gradient" problem with neural networks based on sigmoid non-linearities?

   Answer: When taking the sigmoid of a sigmoid function, the gradient away from the origin is extremely close to zero. So in deep networks, many gradients can be nearly zero everywhere.

10. List 3 forms of regularization we use to prevent overfitting in neural networks.

    Answer: L2 regularization, early stopping, and Dropout.

11. Convolutional networks seem like a pain... why not just use regular ("fully connected") neural networks for image classification

    Answer: The W in fully connected layer is unrestricted, the huge number might cause out of storage and overfitting.