

CPSC 340 Assignment 6 (due Friday November 30th at 11:55pm)

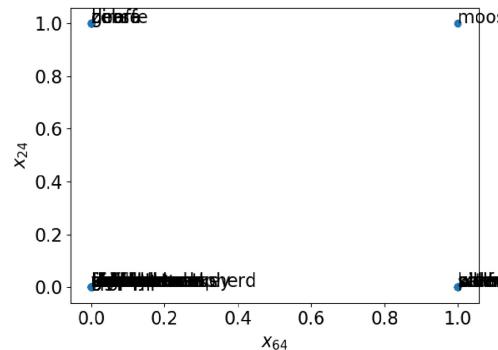
## Instructions

Rubric: {mechanics:5}

The above points are allocated for following the general homework instructions.

## 1 Data Visualization

If you run `python main.py -q 1`, it will load the animals dataset and create a scatterplot based on two randomly selected features. We label some points, but because of the binary features the scatterplot shows us almost nothing about the data. One such scatterplot looks like this:



### 1.1 PCA for visualization

Rubric: {reasoning:2}

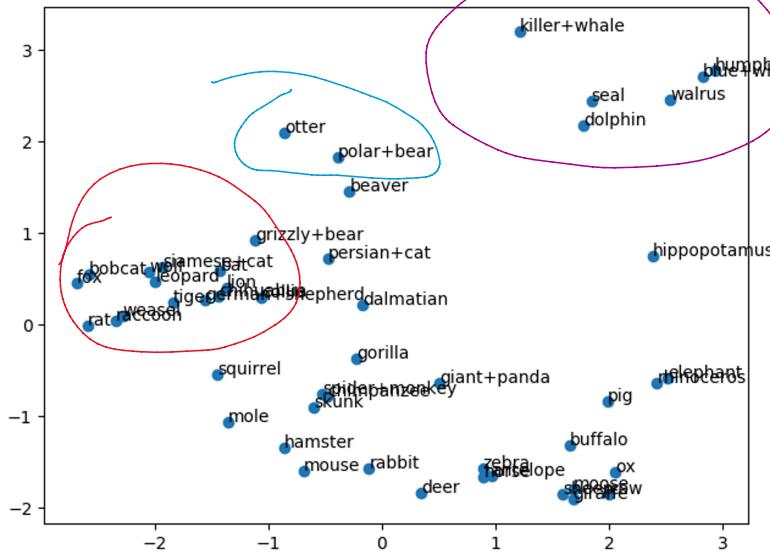
Use scikit-learn's PCA to reduce this 85-dimensional dataset down to 2 dimensions, and plot the result. Briefly comment on the results (just say anything that makes sense and indicates that you actually looked at the plot).

### 1.2 Data Compression

Rubric: {reasoning:2}

1. How much of the variance is explained by our 2-dimensional representation from the previous question?
2. How many PCs are required to explain 50% of the variance in the data?

(1.1)



Felines and sea animals are grouped correctly as shown by red above which sounds right. Yet otter and polar bears are put together which does not sound right

(1.2) (1) For 2 components 32.3%  
the variance is explained.

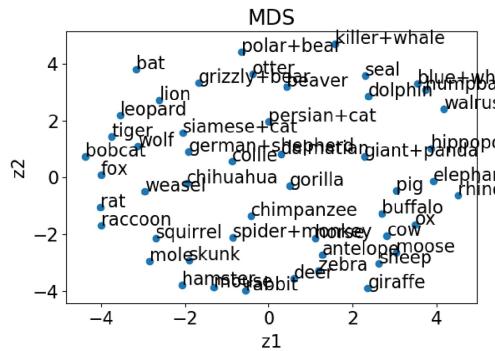
(2) We need 5 components to explain  
50% of the variance. The exact  
percentage of variance  
explained is 54.4 %

### 1.3 Multi-Dimensional Scaling

If you run `python main.py -q 1.3`, the code will load the animals dataset and then apply gradient descent to minimize the following multi-dimensional scaling (MDS) objective (starting from the PCA solution):

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

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. [Compare the MDS objective for the MDS solution vs. the PCA solution; is it indeed lower for the MDS solution?](#)

### 1.4 ISOMAP

Rubric: {code:10}

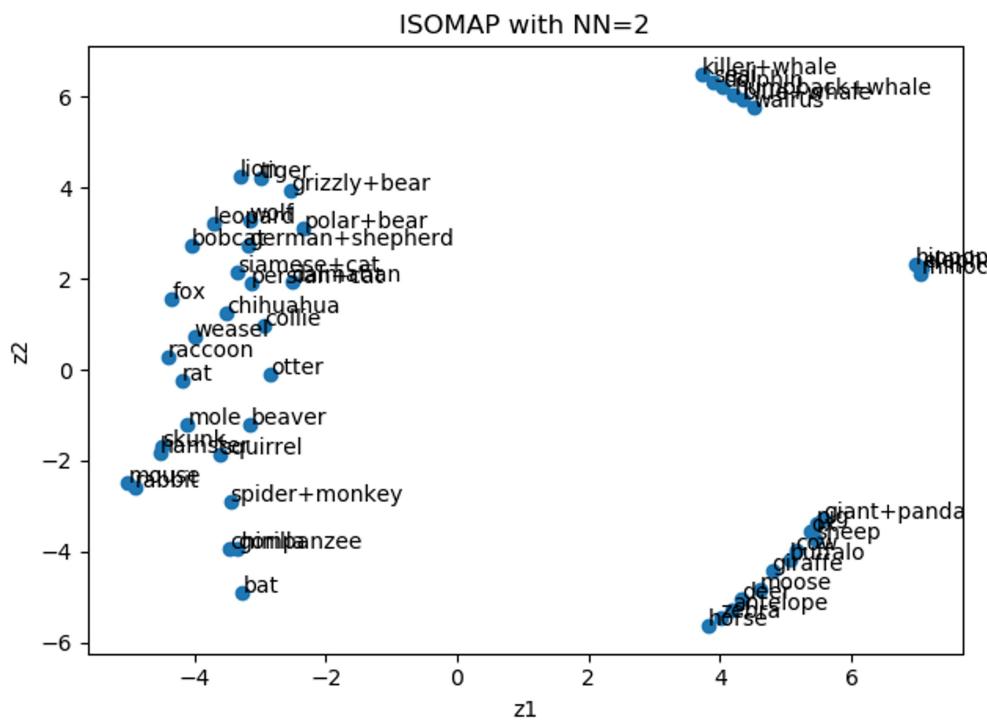
Euclidean distances between very different animals are unlikely to be particularly meaningful. However, since related animals tend to share similar traits we might expect the animals to live on a low-dimensional manifold. This suggests that ISOMAP may give a better visualization. Fill in the class *ISOMAP* so that it computes the approximate geodesic distance (shortest path through a graph where the edges are only between nodes that are  $k$ -nearest neighbours) between each pair of points, and then fits a standard MDS model (1) using gradient descent. [Plot the results using 2 and using 3-nearest neighbours.](#)

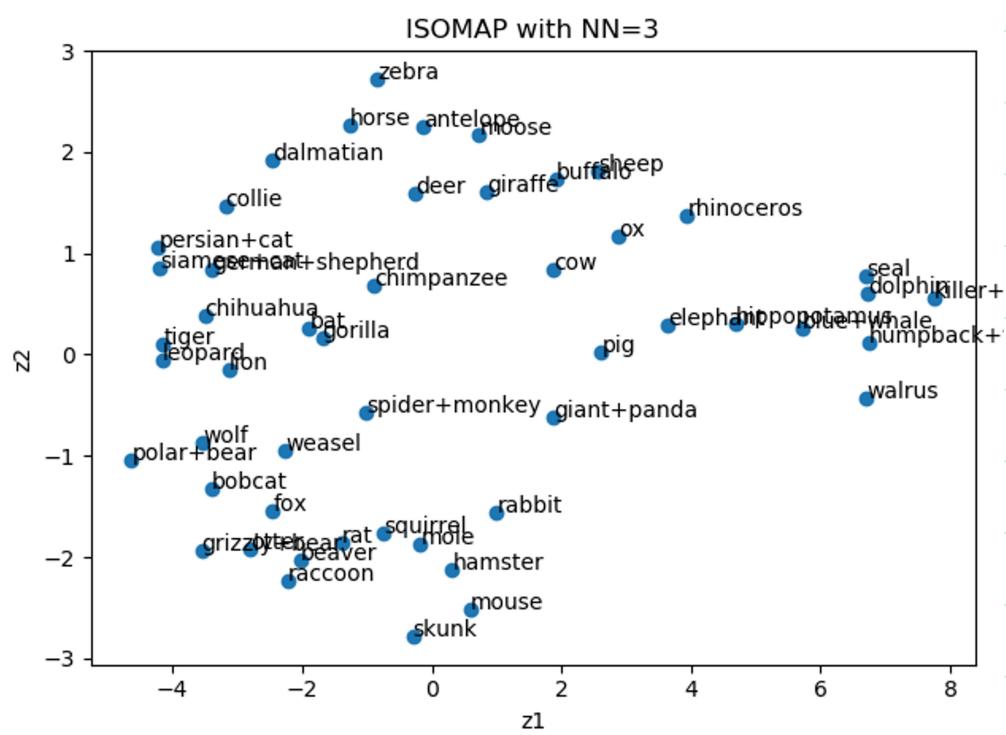
Note: when we say 2 nearest neighbours, we mean the two closest neighbours excluding the point itself. This is the opposite convention from what we used in KNN at the start of the course.

The function `utils.dijkstra` can be used to compute the shortest (weighted) distance between two points in a weighted graph. This function requires an  $n \times n$  matrix giving the weights on each edge (use 0 as the weight for absent edges). Note that ISOMAP uses an undirected graph, while the  $k$ -nearest neighbour graph might be asymmetric. One of the usual heuristics to turn this into a undirected graph is to include an edge  $i$  to  $j$  if  $i$  is a KNN of  $j$  or if  $j$  is a KNN of  $i$ . (Another possibility is to include an edge only if  $i$  and  $j$  are mutually KNNs.)

1776.8183112784675. Error with PCA solution is 4942.195453022984. So MDS solution has a smaller error with the MDS objective

(1.4)





## 1.5 t-SNE

Rubric: {reasoning:1}

Try running scikit-learn's t-SNE on this dataset as well. Submit the plot from running t-SNE. Then, briefly comment on PCA vs. MDS vs. ISOMAP vs. t-SNE for dimensionality reduction on this particular data set. In your opinion, which method did the best job and why?

Note: There is no single correct answer here! Also, please do not write more than 3 sentences.

## 1.6 Sensitivity to Initialization

Rubric: {reasoning:2}

For each of the four methods (PCA, MDS, ISOMAP, t-SNE) tried above, which ones give different results when you re-run the code? Does this match up with what we discussed in lectures, about which methods are sensitive to initialization and which ones aren't? Briefly discuss.

# 2 Neural Networks

NOTE: before starting this question you need to download the MNIST dataset from <http://deeplearning.net/data/mnist/mnist.pkl.gz> and place it in your *data* directory.

## 2.1 Neural Networks by Hand

Rubric: {reasoning:5}

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):

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

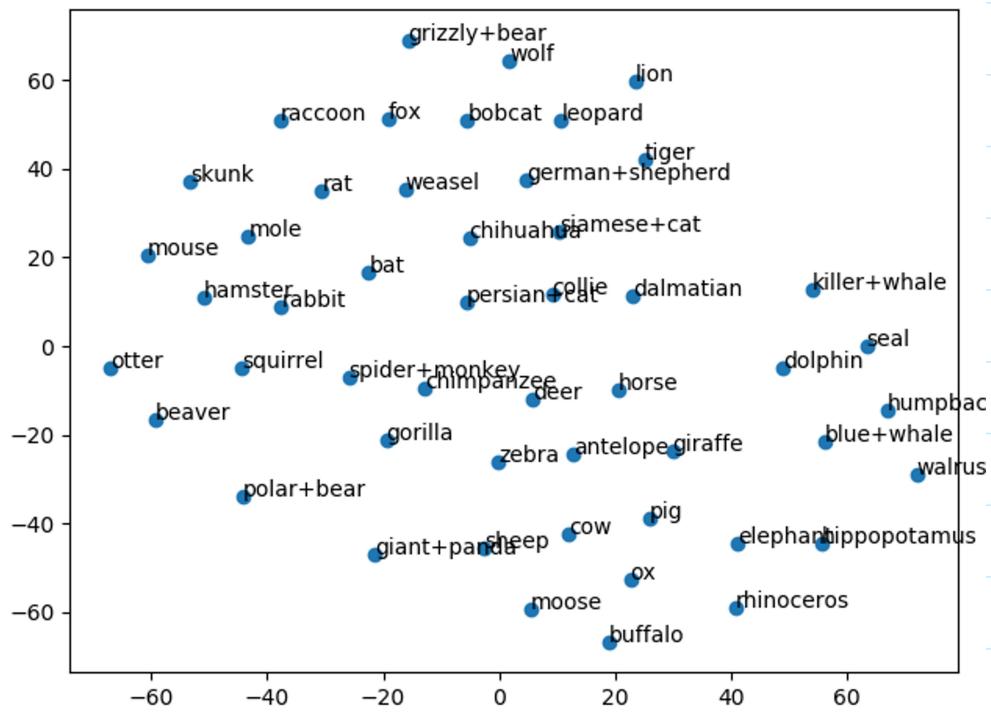
Assuming that we are doing regression, for a training example with features  $x_i^T = [-3 \ -2 \ 2]$  what are the values in this network of  $z_i$ ,  $h(z_i)$ , and  $\hat{y}_i$ ?

## 2.2 SGD for a Neural Network: implementation

Rubric: {code:5}

If you run `python main.py -q 2` it will train a one-hidden-layer neural network on the MNIST handwritten digits data set using the `findMin` gradient descent code from your previous assignments. After running for the default number of gradient descent iterations (100), it tends to get a training and test error of around 5% (depending on the random initialization). Modify the code to instead use stochastic gradient descent. Use a minibatch size of 500 (which is 1% of the training data) and a constant learning rate of  $\alpha = 0.001$ . Report your train/test errors after 10 epochs on the MNIST data.

looks very clean. The distances between animals look meaningful and the clusters are well-defined. Big herbivore mammals are clustered in the bottom, sea animals are on the right, small mammals are on the left and big carnivores are on top.



1.6) Only TSNE changes with initialization, the rest of the algorithms are not sensitive to the initialization. \*\*\* This is in parallel to what we discussed in the class

$$(2.1) z_{i2} w_{x_i} \Rightarrow z_{i2} \begin{bmatrix} -2 & 2 & -1 \\ 1 & -2 & 0 \end{bmatrix} \begin{bmatrix} -3 \\ -2 \\ 2 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 6-4-2 \\ -3+4 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$h(z_i) = \begin{bmatrix} \frac{1}{1+\exp(0)} \\ \frac{1}{1+\exp(-1)} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{1+\frac{1}{e}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{e}{e+1} \end{bmatrix}$$

$$\hat{y}_i = v^T h(z_i)$$

$$= [3 \ 1] \begin{bmatrix} \frac{1}{2} \\ \frac{e}{e+1} \end{bmatrix}$$

$$= \frac{3}{2} + \frac{e}{e+1}$$

(2.2) Fitting took 8 seconds  
 Training error = 0.08466  
 Test error = 0.082

### 2.3 SGD for a Neural Network: discussion

Rubric: {reasoning:1}

Compare the stochastic gradient implementation with the gradient descent implementation for this neural network. Which do you think is better? (There is no single correct answer here!)

### 2.4 Hyperparameter Tuning

Rubric: {reasoning:2}

If you run `python main.py -q 2.4` it will train a neural network on the MNIST data using scikit-learn's neural network implementation (which, incidentally, was written by a former CPSC 340 TA). Using the default hyperparameters, the model achieves a training error of zero (or very tiny), and a test error of around 2%. Try to improve the performance of the neural network by tuning the hyperparameters. Hand in a list changes you tried. Write a couple sentences explaining why you think your changes improved (or didn't improve) the performance. When appropriate, refer to concepts from the course like overfitting or optimization.

For a list of hyperparameters and their definitions, see the scikit-learn `MLPClassifier` documentation: [http://scikit-learn.org/stable/modules/generated/sklearn.neural\\_network.MLPClassifier.html](http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html). Note: "MLP" stands for Multi-Layer Perceptron, which is another name for artificial neural network.

## 3 Very-Short Answer Questions

Rubric: {reasoning:12}

1. Is non-negative least squares convex?
2. Name two reasons you might want sparse solutions with a latent-factor model.
3. Is ISOMAP mainly used for supervised or unsupervised learning? Is it parametric or non-parametric?
4. Which is better for recommending moves to a new user, collaborative filtering or content-based filtering? Briefly justify your answer.
5. Collaborative filtering and PCA both minimizing the squared error when approximating each  $x_{ij}$  by  $\langle w^j, z_i \rangle$ ; what are two differences between them?
6. Are neural networks mainly used for supervised or unsupervised learning? Are they parametric or nonparametric?
7. Why might regularization become more important as we add layers to a neural network?
8. With stochastic gradient descent, the loss might go up or down each time the parameters are updated. However, we don't actually know which of these cases occurred. Explain why it doesn't make sense to check whether the loss went up/down after each update.
9. Consider using a fully-connected neural network for 3-class classification on a problem with  $d = 10$ . If the network has one hidden layer of size  $k = 100$ , how many parameters (including biases), does the network have?
10. The loss for a neural network is typically non-convex. Give one set of hyperparameters for which the loss is actually convex.
11. What is the "vanishing gradient" problem with neural networks based on sigmoid non-linearities?

Fitting took 73 seconds  
Training error = 0.04554  
Test error = 0.0484

I guess in this case since 4% change in the error is somewhat significant I would pick the regular gradient finding method even though it takes 8x more time

2.4) I tried changing learning\_step, learning\_step\_init, batch\_size, tol and max\_iterations. When I increased the max\_iterations and reduced tolerance I expected the test to go up and it did. But when I decreased it, I thought it would reduce overfitting but I still got higher test error. I cannot explain why

3) 1) The NNLS objective is convex because we know that least squares objective is convex, so at any point the function is convex. Taking the positive part preserves the convexity

2) Since we generally want to reduce the number of features we have in a latent factor model, the sparsity reduces the number of features we have. Thus it allows to make predictions cheaper. Also, if our sparsity eliminates negatives then we have increased interpretability

3) ISOMAP is non-parametric and it is mainly used for unsupervised learning because we

are trying to discover what the labels are through solving the manifold

- 4) Collaborative filtering cannot predict on new users. So, content based filtering is better for predicting on new users because we need the user's characteristics in collaborative filtering. However, collaborative filtering can fill in the missing values but PCA cannot
- 5) Collaborative filtering cannot predict on new users whereas PCA can make predictions or transformation on new data points.
- 6) Even though neural nets can both be supervised or unsupervised, they are mainly used for unsupervised learning.  
Neural networks are non-parametric
- 7) Since adding more layers mean more degrees of freedom for learning, we risk overfitting. Thus, introducing regularization reduces this risk
- 8) Because we do not care about the change in loss in each step. We know that in expectation our loss is going to converge to a quasi-optimal solution in function of our learning step. So, checking the change in loss in each step is a waste of time
- 9) The connection between the input layer and the hidden layer has  $10 \times 100 = 1000$

parameters. Combining with the biases, we will have 1100 parameters between the input and hidden layers. The connection between the hidden layer and output has  $100 \times 3 + 3 = 303$  parameters. Therefore in total we have 1403 parameters

10) If we use 'relu' activation and one hidden layer then the neural network objective function is convex

11) Since the derivative approaches zero very quickly when moving away from 0 when we use sigmoid function, when finding the gradient the values of gradient approaches zero causing underflow and overflow issues thus "vanishing gradient"

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

12) We use convolutional networks because when we are making predictions based on an image, we would like to make a prediction using the neighboring pixels in the image not by the pixel itself. Thus, using the convolution we are able to capture more complex patterns in the data

