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

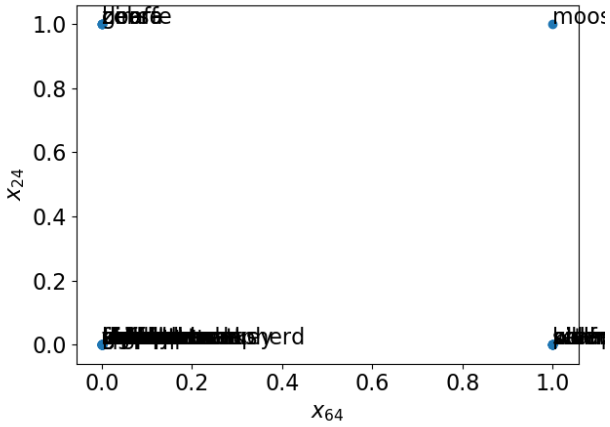
Instructions

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}

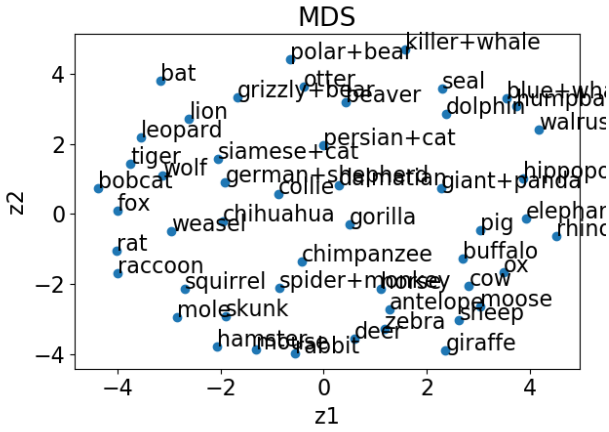
1. How much of the variance is explained by our 2-dimensional representation from the previous question?
2. How many PCs are required to explain 50% of the variance in the data?

1.3 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):

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

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. 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 neigbours excluding the point itself. This is the opposite convention from what we used in KNN at the start of the course.

The function *utils.dijskstra* 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.)

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 $x_i^T = \begin{bmatrix} -3 & -2 & 2 \end{bmatrix}$ 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.

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!)

Answer It is significantly faster with the stochastic gradient descent, therefore this is better in terms of the computational cost, especially for computationally expensive neural network architecture. However, based on the results in 2.2, gradient descent gives smaller training/test error, therefore it is better in terms of the prediction performance.

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 hyperparemeters. 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`. Note: "MLP" stands for Multi-Layer Perceptron, which is another name for artificial neural network.

hidden_layer_sizes	alpha	solver	early_stopping	activation	Training error	Test error
				logistic	0.0	0.0223
200					0.0	0.0183
200	0				0.0	0.02
200	0.001				0.0	0.018
200	0.01				0.00004	0.0191
200			True		0.002	0.0196
300					0.0	0.0164
500					0.0	0.017
500	0.01				0.0	0.0154
1000					0.00784	0.0239
1000	0.01				0.00002	0.0148

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

Answer

1. Yes. Least square is convex everywhere, therefore the subset of it is also convex.
2. Each latent-factor can be explained by a small number of neurons. Each neuron tend to be human-interpretable parts of the object.
3. It is used for supervised learning and it is non-parametric.
4. Content-based filtering is better for a new user, because collaborative filtering cannot be applied to be a new user.
5. Collaborative filtering only works on the available entries of Y.
6. Neural networks are used for supervised learnings and they are non-parametric.
7. Because as the number of layers increases, the model becomes more complex and they would overfit to the training sets.
8. Because average of the gradients is the full gradient and on average the algorithm should go in the right direction.
9. Bias : 1, w : 10, z : 100, v : 100, therefore, 211 parameters.
10. Linear transformation as a activation function (?)
11. Gradient is nearly zero everywhere away from the origin.
12. Because it is difficult to take into account information from neighbouring pixels with regular neural networks.