DSC 257R: Unsupervised learning

Homework 1

Proximity in data spaces

1. In each of the following situations, specify the data space $\mathcal{X}$ using correct mathematical notation.
   (a) Each data point is a 10-dimensional vector with real-valued entries.
   (b) Each data point is a 3-dimensional vector with entries in $[0, 1]$.

2. Determine the Euclidean (ℓ_2) distance between each of the following pairs of points p and q.
   (a) $p = 1$, $q = 10$
   (b) $p = \begin{bmatrix} -1 \\ 12 \end{bmatrix}$, $q = \begin{bmatrix} 6 \\ -12 \end{bmatrix}$
   (c) $p = \begin{bmatrix} 1 \\ 5 \\ -1 \end{bmatrix}$, $q = \begin{bmatrix} 5 \\ 2 \\ 11 \end{bmatrix}$

3. Consider the vector $x = \begin{bmatrix} 10 \\ 15 \\ 25 \end{bmatrix}$.
   (a) Let p the result of *scaling* this vector (that is, multiplying all its entries by the same factor) so that the sum of the entries is 1. What is p?
   (b) This vector p lies in the probability simplex Δ_k for what k?

4. Give an example of a two-dimensional point that *cannot* be scaled to lie in Δ_2.

5. Sketch the probability simplex Δ_3 using three-dimensional axes. Make sure to label the axes. Show the locations of the following points:

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

   Also label the most central point in the simplex; what are its coordinates?

6. Here are three probability distributions in Δ_4.

$$p = \begin{bmatrix} 1/2 \\ 1/4 \\ 1/8 \\ 1/8 \end{bmatrix}, \quad q = \begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}, \quad r = \begin{bmatrix} 1/2 \\ 0 \\ 1/4 \\ 1/4 \end{bmatrix},$$

   (a) Compute ℓ_1 distance between p and q, that is, $\|p - q\|_1$.

(b) Compute $\|q - r\|_1$.

(c) Compute the KL divergence $K(p, q)$.

(d) Compute the KL divergence $K(q, r)$.

Before attempting the next problems, make sure that Python 3 and Jupyter are installed on your computer.

7. *Choosing representations for nearest neighbor.* In this problem, we will study how different representations of images can affect the performance of nearest neighbor methods. We will use the CIFAR-10 data set, which has 50,000 training images and 10,000 test images, with ten different classes (`airplane`, `automobile`, `bird`, `cat`, `deer`, `dog`, `frog`, `horse`, `ship`, `truck`). The images are in color, of size 32×32.

   We will compare several image representations:

   - The raw pixel representation
   - Histogram-of-gradients (HoG) features
   - The representation obtained by passing the image through a pre-trained convolutional net (VGG) and using one of the last layers (`last-fc`, meaning "last fully-connected layer")
   - The representation obtained by passing the image through a pre-trained convolutional net (VGG) and using one of the earlier layers (`last-conv`, meaning "last convolutional layer")
   - The representation obtained by using a convolutional net with the same architecture but with *random weights* (and again, with two variants, `last-fc` and `last-conv`)

   In each case, the idea is study the classification performance (on the test set) using 1-nearest neighbor on the training data with Euclidean (ℓ_2) distance.

   Download `cifar-representations.zip` from the course website. The directory contains a Jupyter notebook, some helper functions, and some data. In the notebook, we have provided code that will extract HOG and neural net features from the CIFAR data; look through it to get a sense of how it works.

   (a) What is the dimensionality of each of the representations (raw pixel, HoG, VGG-last-fc, VGG-last-conv)?

   (b) Report test accuracies for 1-nearest neighbor classification using the various representations (raw pixel, HoG, VGG-last-fc, VGG-last-conv, random-VGG-last-fc, random-VGG-last-conv).

   (c) For the raw pixel representation:

   - Show the first five images in the test set whose label is *correctly* predicted by 1-NN, and show the nearest neighbor (in the training set) of each of these images.
   - Show the first five images in the test set whose label is *incorrectly* predicted by 1-NN, and show the nearest neighbor (in the training set) of each of the images.

   Repeat for the HoG and VGG-last-fc representations.

8. *Word vectors.* In this problem, we'll get a first glimpse of *word embeddings*. We will work with GloVe, which provides 300-dimensional vectors for each of 400,000 words; you can find out more about these at `https://nlp.stanford.edu/projects/glove/`.

   The vectors are in the file `glove.6B.300d.txt`, which you can download from the course website. To load them into Python, use the following code:

```
import numpy as np
filename = 'glove.6B.300d.txt'
with open(filename) as f:
   content = f.read().splitlines()
n = len(content)
vecs = np.zeros((n,300))
words = []
index = 0
for rawline in content:
   line = rawline.split()
   words.append(line[0])
   vecs[index] = np.genfromtxt(line[1:])
   index = index+1
```

Once this has run, the following variables will be set:

- `n`, the vocabulary size
- `words[0:n]`, a list of the vocabulary words
- `vecs[0:n]`, the word vectors

The following words are all in the vocabulary:

'communism', 'africa', 'happy', 'sad', 'upset', 'computer', 'cat', 'dollar'

For each of these words:

- Find the five closest words (other than itself) in the 300-dimensional embedding space. (Either code up your own nearest neighbor algorithm or use `sklearn.neighbors.NearestNeighbors`.)
- Turn in these lists of neighboring words.