CS 131 Computer Vision: Foundations and Applications (Fall 2014) Problem Set 3

Due Wednesday, December 3, 5 PM

1 Curse of Dimensionality

In class, we used the nearest neighbor method to perform face recognition. The nearest neighbor method is intuitive and very powerful, especially in low dimensional applications. When the number of dimensions becomes higher, however, we may not want to use this method directly because of its computational overhead and counterintuitive geometry. To work around these limitations, we introduced PCA to reduce the dimensions of the data.

(a) Suppose we have n data samples and w test points in d dimensional space, we want to find each test point's nearest neighbor based on Euclidean distance $\|\cdot\|_2$ where

$$\|x-y\|_2 = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$$

Please give the computational complexity of this procedure using *big-O* notation, and discuss what happens when d becomes very large.

(b) When performing nearest neighbor search using Euclidean distance, we can use a simple heuristic to speed up the search: we quickly reject candidates by using the difference in one dimension as a lower bound for distances. Explain in words why this heuristic will not help much in high dimensional space (i.e. when the dimension d is large).

(c) Given a unit cube in d-dimensional space $C = \{x \in \mathbb{R}^d | 0 \leq x_i \leq 1, i = 1, 2, \ldots, d\}$, show that nearly all the volume of C is concentrated near its surface when d is large. *Hint:* To get the volume of a cube near its surface, you can shrink the cube from its center by some constant c and subtract the volume of shrunk cube from the volume of the original one.

(d) Consider generating points uniformly at random on the surface of a unit-radius sphere in d dimensions. To get an intuition, let's first consider the 2-dimensional version of generating points on the circumference of a unit-radius circle by the following method:

- Independently generate each coordinate uniformly at random from the interval $[-1, 1]$. This produces points distributed over a square that is large enough to completely contain the unit circle.

- Discard all points outside the unit circle and project the remaining points onto circle (by normalizing them into unit 2d vectors).

Can we generalize this technique in the obvious way to higher dimensions? What will happen when d becomes very large? Use your result from part (c) to explain.

(e) Please list at least two methods you could use to overcome problems with high-resolution images (i.e. data points in very high dimensions).

2 1-NN with Different Distance Metrics

Given two vectors x and y in n-dimensional space $\mathbb{R}^n$, we can measure distances between them in multiple ways:

- Eulidean distance (or ℓ_2 distance)

$$Dist^{\text{Euc}}(x, y) = \|x - y\|_2 = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- Cosine distance

$$Dist^{\text{Cos}}(x, y) = 1 - \frac{x^T y}{\|x\|_2 \|y\|_2}$$

- Mahalanobis distance

$$Dist^{\text{Mah}}(x, y, \Sigma) = (x - y)^T \Sigma (x - y)$$

(a) Let Σ be an identity matrix. Prove that:

$$Dist^{\text{Mah}}(x, y, \Sigma) = \|x - y\|_2^2$$

for any x and y. The right hand side of equation is the squared Euclidean distance.

(b) Let $x_0, \ldots, x_K \in \mathbb{R}^n$. Consider the set of points that are closer (in Euclidean distance) to x_0 than the other x_i, i.e.,

$$V = \left\{ x \in \mathbb{R}^n \middle| \|x - x_0\|_2 \leq \|x - x_i\|_2, \quad i = 1, \cdots, K \right\} \tag{1}$$

Show that the set V can be described as $\left\{ x | Ax \preceq b \right\}$, where $Ax \preceq b$ denotes:

$$\begin{aligned} a_{11}x_1 + \cdots + a_{1n}x_n &\leq b_1 \\ a_{21}x_1 + \cdots + a_{2n}x_n &\leq b_2 \\ \vdots \qquad\qquad &\leq \vdots \\ a_{K1}x_1 + \cdots + a_{Kn}x_n &\leq b_K \end{aligned}$$

In your answer, please make sure to specify what A and b are.

(c) Continuing from part (b), the set V defined in equation (1) is called the *Voronoi region* around x_0 with respect to $x_1, \ldots, x_K$. Let's consider the reverse problem of part (b). Given a non-empty set $V = \left\{x | Ax \preceq b\right\}$, describe in words how to find $x_0, \ldots, x_K$ so that V is the Voronoi region of x_0 with respect to $x_1, \ldots, x_K$.

Hint: The goal is not to find the original, unique points that produced the Voronoi set V. That problem would be under-constrained (fewer equations than unknowns). Instead, all you need to do is describe how to find one possible set of points $x_0, \ldots, x_K$ that will produce that same Voronoi region around x_0 (it won't necessarily produce the original Voronoi regions around the other points).

(d) Using Euclidean distance, we can define a family of sets

$$V_k = \left\{x \in \mathbb{R}^n \middle| \|x - x_k\|_2 \leq \|x - x_i\|_2, \quad i \neq k\right\}$$

The set V_k consists of points in $\mathbb{R}^n$ for which the closest point in the set $\{x_0, \ldots, x_K\}$ is x_k. The family of sets $V_0, \ldots, V_K$ gives a decomposition of the space $\mathbb{R}^n$.

Using Cosine distance, we can also define a family of sets

$$W_k = \left\{x \in \mathbb{R}^n \middle| Dist^{\text{Cos}}(x, x_k) \leq Dist^{\text{Cos}}(x, x_i), \quad i \neq k\right\}$$

Assume $x_0, \ldots, x_K$ are unit vectors. Prove that the decompositions $V_0, \ldots V_K$ and $W_0, \ldots, W_K$ are equivalent. Two decompositions of $\mathbb{R}^n$ are equivalent if and only if for any $a \in \mathbb{R}^n$,

$$a \in V_k \iff a \in W_k$$

(e) Assume, again, that $x_0, \ldots, x_K$ are unit vectors. Would choosing Euclidean 1-NN over Cosine 1-NN make a difference? Please justify your answer.

(f) Consider the following in two dimensional space:

$$x_0 = (0, 3) \quad x_1 = (-3, 0) \quad x_2 = (3, 0) \qquad \Sigma = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

In particular, consider the decomposition U_0, U_1 and U_2 given by

$$U_k = \left\{x \in \mathbb{R}^n \middle| Dist^{\text{Mah}}(x, x_k) \leq Dist^{\text{Mah}}(x, x_i), \quad i \neq k\right\}$$

where $k = 1, 2, 3$. Use Matlab to plot this decomposition. Submit your code and plot for this part of the problem. *Hint:* To plot a decomposition, you only need to plot the "decision boundaries" between set U_i and set U_j.

3 PCA as Fitting Lines

PCA projects the data by an orthogonal matrix. This problem shows how to obtain this orthogonal matrix, as well as the intuition behind it. For simplicity, you are only required to derive PCA in two

dimensional space. After finishing this problem, you can easily extend the results to n-dimensional spaces.

Suppose we have a dataset $\mathcal{D} \in \mathbb{R}^{m\times 2}$ in $\mathbb{R}^2$, where each data sample $x^{(i)}$ is one row of the data matrix. We want to approximate our high dimensional data by an orthogonal line set L in two dimensional space

$$L = \{az + b | z \in \mathbb{R}\}$$

where $a \in \mathbb{R}^2$, $b \in \mathbb{R}^2$, and a is a unit vector.

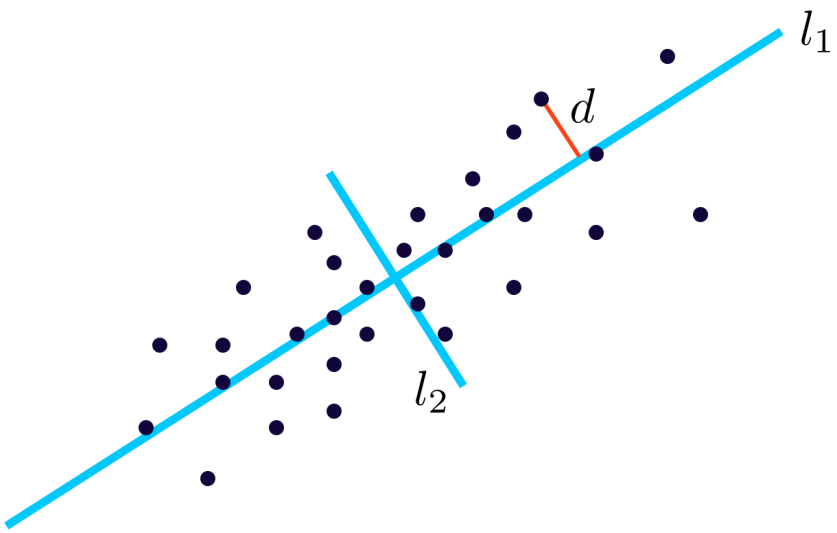


Figure 1: An illustration of fitting a two dimensional dataset to a line set $L = \{l_1, l_2\}$. Each black dot is a data sample.

For each data sample $x^{(i)}$, the distance between x and a line l is defined by

$$d(x^{(i)}, l) = \min_{y\in l} \|x^{(i)} - y\|_2,$$

where $\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$ (ℓ_2 norm), n is the data dimension and it is 2 in our case. Since we have m data samples, we start by finding a single line set l that minimizes the total squared distance:

$$d_t = \sum_{i=1}^{m} d(x^{(i)}, l)^2$$

In other words, we'd like to fit the data samples to a set of straight lines, as close as possible such that d_t is minimized.

(a) Show that $d(x, l) = \|(I - aa^T)(x - b)\|_2$. *Hint: you might consider using a least squares approach.*

(b) Using your result from (a), the total squared distance is given by

$$d_t = \sum_{i=1}^{m} \|(I - aa^T)(x^{(i)} - b)\|_2^2$$

Show that the optimal b, which minimizes d_t while all other variables are held constant, is given by

$$b_{\text{opt}} = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$$

Here, b_{opt} is the mean (center) of our data set. *Hint:* $P = (I - aa^T)$ *is a projection matrix.*

(c) Using this choice of $b = b_{\text{opt}}$, find the optimal value of a which minimizes d_t, in terms of $x^{(i)}$ and b_{opt}. *Hint:* Try to formulate the problem as a maximization problem of the form $\max(a^T \Sigma a)$. The linear algebra review notes from the first week of class may prove useful here.

(d) So far, our results only give us a single line $l = a_{\text{opt}} z + b_{\text{opt}}$ ($z \in \mathbb{R}$). Our final goal is to approximate the 2D data by an *orthogonal* line set L in two dimensional space. Describe how you would choose the other line to add to the set L. *Hint:* The procedure should be very similar to the steps in part (a)-(c).

4 Dimensionality Reduction

Because of the "curse of dimensionality", we usually want to reduce our data dimensions when comparing images. We introduced Principle Component Analysis (PCA) as a method for dimensionality reduction. In this problem, you're going to learn another two common methods of dimensionality reduction, which are also widely used in practice:

- **Image downsampling** is a common way of making images smaller. Matlab's `imresize` does image downsampling (its default options are fine for our purposes).
- **Random projection** uses randomly-chosen basis vectors to project high-dimensional data points into a lower-dimensional space.

  Suppose we have m data points $\{d_1, \ldots, d_m\}$ in $\mathbb{R}^n$, where n is a very large number. We project each d_i by multiplying a randomly generated matrix R on the left side, i.e. Rd_i. We deliberately choose R to be "flat", i.e. of size $s \times n$ where $s \ll n$ such that the projection Rd_i is of a lower dimension. We obtain R by randomly generating each of its entries independently from a standard Gaussian distribution $\mathcal{N}(0, 1)$, and normalize each row to be a unit vector. In this problem, we have generated the random vectors for you.

  There are some nice theoretical properties about random projection, such as the Johnson-Lindenstrauss lemma, which is beyond the scope of CS131.
- **Principal component analysis** projects data along the directions of its largest variances. For this problem, we have provided the principle components in `component.mat`, where each of row of `U` is a principal component.

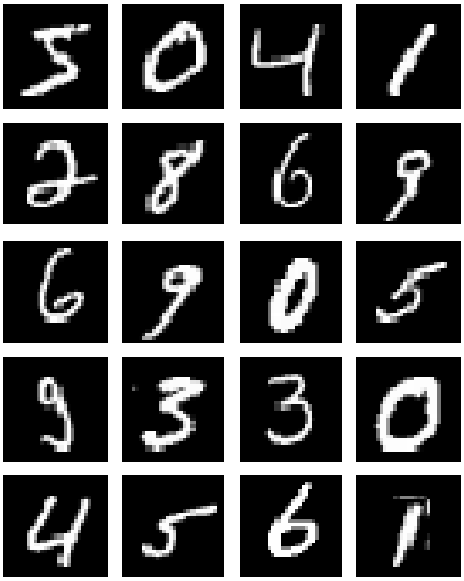


Figure 2: Handwritten digits. Our KNN classifier will compare new images to known training images.

(a) We will use the famous MNIST handwritten digits dataset [2] for this problem. Figure 2 is a snapshot of the dataset. Each gray-scale image is of size 28×28, which contains 784 pixels. We have subsampled the original dataset into `Train.mat` and `Test.mat`.

> `Train.mat` is cell array of length 10. The cell at position i (assume we index starting from 1) contains a 784×500 matrix, which contains 500 images for digit `(i mod 10)`.
>
> `Test.mat` contains X and Y, where Y contains labels and X contains data.

We also provide an implementation of K-nearest-neighbors algorithm in `KNN.m`. For each test instance, we compute its Euclidean distances to each data point in the `Train` cell array and decide the test instance label by a voting procedure of class labels of its K nearest neighbors. In this problem, you should implement `DownSample.m`, `RandomProj.m` and `PCA.m` to reduce the dimensionality of the data. After you finish, you should run `KNN.m` and report classification performance with each of the three dimensionality-reduction methods. Please submit a printout of your code.

(b) What do you observe when varying the number of reduced dimensions? Plot a (classification accuracy) vs. (number of dimensions) curve for each of the three reduction methods (i.e. we need three curves total). For `Downsample`, please use downsample ratios 0.1, 0.2, 0.4, and 0.8. For `RandomProj` and `PCA`, please use reduced dimensions 9, 36, 144, and 576.

(c) Are the results you obtained in parts (a) and (b) sensitive to the number of nearest neighbors, K? Please choose one of the three reduction methods and plot ONE (classification accuracy) vs. (K) curve. We recommend using $K = 5$, 10, 20, and 40.

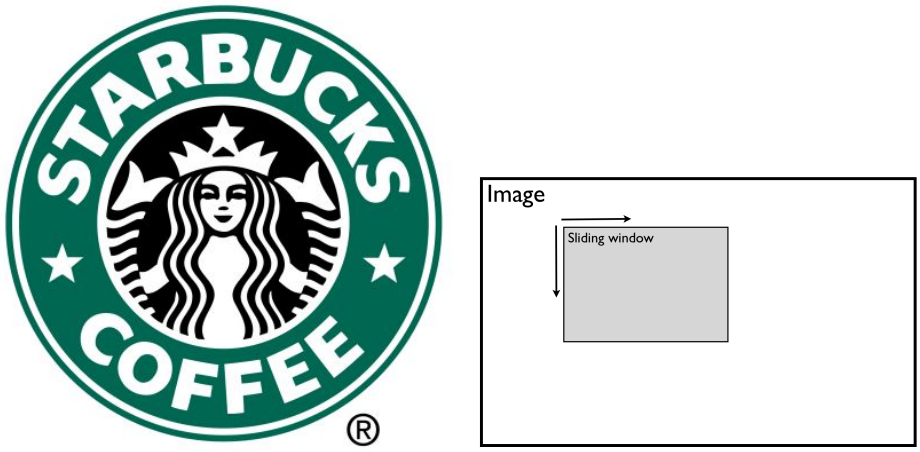
(d) Please rank these three dimensionality reduction methods with respect to their running time, accuracy in part (b), and robustness in Table 1. Notice "robustness" column has been filled for you. When your input data is noisy, random projection is the most robust among these three. Image downsampling ranks second, and PCA is the least robust reduction method.

Reduction methods	Running time in part(b)	Recognition accuracy in part(b)	Robustness
Image downsamping			2
Random projection			1
PCA			3

Table 1: Comparison of three reduction methods.

5 Logo Detection

A lot of computer vision algorithms represent visual data in the form of histograms (such as SIFT, Shape Context), which can be easily handled by computers but allow for "slop" in the exact pixel values and locations. In this problem, we are going to develop a simple detector for Starbucks logos using histograms of image colors, and we'll use a sliding window search to apply our detector to an image.



(a) A template Starbucks logo (b) Sliding window search

Figure 3: Problem 5 illustration.

Suppose we have a template Starbucks logo, as given in Figure 3(a). For some given images (under the `logo/data` folder), we want to find regions that are visually similar (i.e. likely another Starbucks logo).

We'll use a sliding window search as shown in Figure 3(b), where we slide a fixed size rectangle across the entire image and compare the contents of each rectangle with the given template image.

We repeat the process with multiple window sizes. The sliding window code is provided for you; in this problem you will only need to handle the comparison.

We will use color histograms to compare the template to each image rectangle that is grabbed by the sliding window. We'll build a histogram for the image inside the window, compare it against the histogram of the template, and record the sliding window position which yields the best match.

Here is a list of functions you will use:

- `Histogram.m` builds a color histogram from a given image patch.
- `SlideWindowDetector.m` compares the template to all possible image regions, and saves the one with the best match.
- `VisualizeBox.m` visualizes the detection result from `SlideWindowDetector.m`.

(a) In this part, you need to implement two histogram measures and use these two measures to find the best match to the Starbucks logo in several images.

- Histogram intersection distance

$$Dist^{\text{Intersection}}(x, y) = \sum_{i=1}^{n} \min\{x_i, y_i\}$$

  This distance is a similarity measure, i.e. the higher the distance is, the more similar two histograms are.

- Chi-square distance

$$Dist^{\text{Chi-square}}(x, y) = \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{x_i + y_i}$$

  This distance measures the proximity of two histograms, i.e. the lower the value is, the more similar two histograms are. Note: for entries where the denominator above is 0, just add 0.

Complete `HistIntersectDist.m` and `ChiSquareDist.m`, and then test your code by running `LogoRecognition.m` after uncommenting the relevant function call. Please submit printout of your code for this part of the problem.

(b) From part (a), we can see that our algorithm is not very accurate. This is partly because a color histogram of an entire image is a quite weak representation (for example, it discards all the spatial information). In this part, you are going to incorporate spatial information into the color histogram by implementing "Spatial Pyramid"[1].

Read section 3 of the included "Spatial Pyramid" paper [1], and implement its equation (3) in `SpatialPyramidDist.m`. The basic idea is to compute color histogram distances between multiple **sub-regions** of the images being tested, so that spatial information is not entirely discarded (e.g. when our sliding window is looking for images of the Starbucks logo, the images will need black pixels near the center, rather than just anywhere). Figure 4 illustrates the histogram-building process.

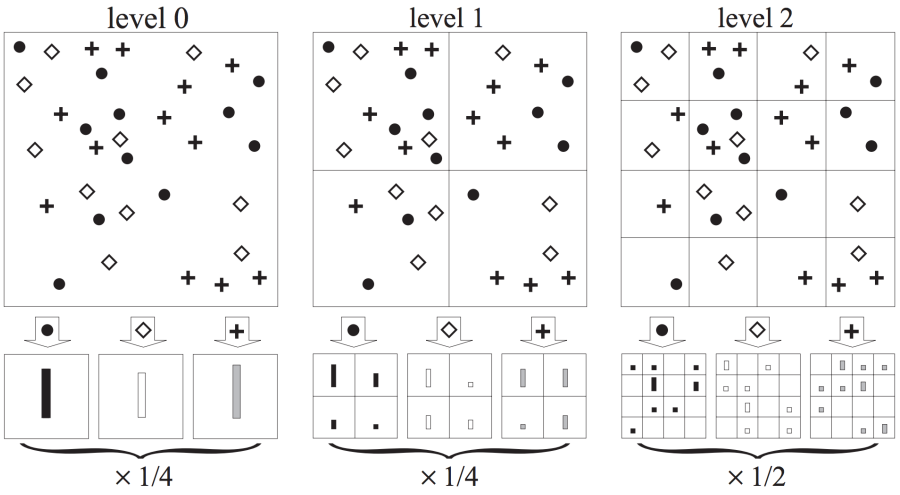


Figure 4: Spatial pyramid

Please submit a printout of your code and an image showing a successful logo detection.

(c) Of the three histogram-based image-comparison methods above, which are invariant to rotations? Explain your answer.

References

[1] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. IEEE, 2006.

[2] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.