CIS 6930/4930 – Assignment 1
Linear Approximation

Due on September 14, 2016, at the *beginning* of class

- This assignment helps us advance our understanding of vectors beyond a vertical array of numbers. It will also help us see how the approximation problem, we learned in class, applies in practice (e.g., JPEG).
- Please submit your homework in hard copy. For computer problems, submit a written report documenting your experiments. Moreover, you need to submit a printout of the source code you used for running the experiments. If you discuss exercises/problems with others or use resources for any part of the assignment (including ideas, source code), make sure to include proper citation and clear acknowledgments in your report.
- Ask lots of questions! Make use of the "Discussion" forum on canvas. Asking questions as well as helping others count as class participation.
- Late submissions are penalized by 10% of the grade for each day (maximum of four) past the due date.

1 Polynomial Approximation

The objects we are interested in this exercise are the continuous functions over the $[0,1]$ domain. In the language we developed in class, we have a vector space $\mathcal{S} = C([0,1])$. We know that $x(t) = e^t$ is an element in $\mathcal{S}$. But since in a finite computational model we often have access to the basic arithmetic operators, we can only afford computing with polynomials (for which you only need arithmetic operators). From the computational costs viewpoint, we can only afford low order polynomials, say quadratic polynomials. This particular (three dimensional) subspace $\mathcal{T}$ of $\mathcal{S}$ is spanned by $\{v_0, v_1, v_2\}$ where $v_0 = 1$ (i.e., a function that is constant 1 on the $[0,1]$), $v_1 = t$ and $v_2 = t^2$.

a. Find $\hat{x}$, the closest point in $\mathcal{T}$, to x when the notion of distance is defined in this norm:

$$\|x - y\|^2 = \int_0^1 (x(t) - y(t))^2 \, \mathrm{d}t \tag{1}$$

In other words, find *the monomial coefficients* a_0, a_1 and a_2 that makes $\hat{x}(t) = a_0 v_0 + a_1 v_1 + a_2 v_2$ is closest to x in this norm. Hint: Consider what inner product induces the above norm. You can use MATLAB tools for integration (i.e., `quad`) and solving linear systems (e.g., `linsolve, inv`...).

b. Calculate the error by computing $\|x - \hat{x}\|$. Also as discussed in class, Taylor series truncation may be considered as an option: $\hat{x}_{\text{Taylor}}(t) = 1 + t + t^2/2$. How does $\|x - \hat{x}_{\text{Taylor}}\|$ compare to the former approximation error? Plot $x, \hat{x}$ and $\hat{x}_{\text{Taylor}}$ on the same axis.

c. Now repeat this exercise for the case when the distance (e.g., length of the error vector) is defined according to this norm:

$$\|x - y\|^2 = \int_0^1 w(t) \left(x(t) - y(t)\right)^2 \mathrm{d}t \qquad \text{where} \quad w(t) = 16(t - 1/2)^2. \tag{2}$$

d. Now plot $x(t)$ and $\hat{x}(t)$ and comment on the difference this norm makes.

2 Trigonometric Approximation

A close relative of the Fourier transform is the Cosine transform which is a convenient choice for dealing with real-valued functions, e.g., sound (1D), image (2D). Specifically, any function $x(t)$ (in $\mathcal{S} = L_2$) is represented by *coefficients* $a_0, a_1, \ldots$ in the cosine expansion:

$$x(t) = a_0 + \sum_{n=1}^{\infty} a_n \cos(\pi n t). \tag{3}$$

The set of cosines $v_n = \cos(\pi n t)$ for $n \geq 1$ can be considered as an orthonormal basis for $\mathcal{S}$ since:

$$\langle v_n, v_{n'} \rangle = 2 \int_0^1 \cos(\pi n t) \cos(\pi n' t) \mathrm{d}t = \begin{cases} 1 & n = n' \\ 0 & n \neq n' \end{cases}. \tag{4}$$

Just like the Discrete Fourier Transform (DFT), the Discrete Cosine Transform (DCT) works in vectors of finite dimension. Discretizing $[0,1]$ by N points allows us to represent $x(t)$ (i.e., a function defined over $t \in [0,1]$) with an N-dimensional vector $\mathbf{x} = [x_1, x_2, \ldots, x_N]^T$ with the index notation $\mathbf{x}(i) = x_i$ for $i = 1, \ldots, N$. Then the DCT basis vectors $\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{N-1} \in \mathbb{R}^N$ form a basis for this space:

$$\mathbf{v}_0(i) = \sqrt{1/N}, \quad \text{and} \quad \mathbf{v}_n(i) = \sqrt{2/N}\cos\left(\pi n \frac{i-1/2}{N}\right). \tag{5}$$

Verify that the DCT basis vectors are orthonormal with regards to the standard inner product:

$$\langle \mathbf{v}_n, \mathbf{v}_{n'} \rangle = \begin{cases} 1 & n = n' \\ 0 & n \neq n' \end{cases}. \tag{6}$$

a. Describe the procedure you would use to provide the best linear approximation to a given vector $\mathbf{x}$ from a k-dimensional subspace $\mathcal{T}$ spanned by *the lowest k frequencies*: $\mathcal{T} = \text{span}(\{\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{k-1}\})$ when k is a number (in the previous exercise it was 3) much smaller than N.

b. An image is a 2D function that when discretized is represented on a 2D array. For instance in MATLAB you can read an (square-sized) image:
`x = imread('AGrayScaleImage.png');` resulting in `x` which is an $N \times N$ array:

$$\mathbf{x} = \begin{matrix} x_{1,1} & x_{1,2} & \cdots & x_{1,N} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,1} & x_{N,2} & \cdots & x_{N,N} \end{matrix}$$

Here is the key point: Even though we are dealing with 2D arrays, we still think of an image as a vector in a vector space. We can add two such vectors, scale them with a number and so on. The standard inner product in this vector space is, therefore, given by:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{N} \sum_{j=1}^{N} \mathbf{x}(i,j)\mathbf{y}(i,j). \tag{7}$$

Pick a set of (gray-scale) images with the same resolution: some visually similar to each other, and some very different. Compute the angle between every pair from your chosen set. What does the angle say about the visual similarity and differences in your set?

c. We can build a set of 2D DCT basis vectors $\boldsymbol{\varphi}_{m,n}$ (with $0 \leq m, n \leq N-1$ resulting in a total of N^2 vectors), by a product of 1D DCT basis vectors:

$$\boldsymbol{\varphi}_{m,n}(i,j) = \mathbf{v}_m(i)\mathbf{v}_n(j). \tag{8}$$

Show the images of few low-frequency basis vectors (e.g., $0 \leq m, n \leq 3$) for an small image size (e.g., N=8, or 16). Verify that the 2D DCT basis vectors are orthonormal:

$$\left\langle \boldsymbol{\varphi}_{m,n}, \boldsymbol{\varphi}_{m',n'} \right\rangle = \begin{cases} 1 & n = n' \text{ and } m = m' \\ 0 & n \neq n' \text{ or } m \neq m' \end{cases}. \tag{9}$$

d. Now given an image $\mathbf{x}$ how can we build low dimensional approximations? An image $\mathbf{x}$ with a resolution of $N \times N$ is represented by N^2 numbers (i.e., pixels). If we have a budget (e.g., number of bytes we are allowed to use) and want to get an image with a fraction of that data, throwing away pixels would not be such a bright idea (which pixels to throw away?). Instead, we can build a low dimensional subspace $\mathcal{T}$, using 2D DCT basis vectors, and find $\hat{\mathbf{x}}_k$ – the best linear approximation to $\mathbf{x}$ from the k-dimensional $\mathcal{T}$. In general given a budget k, we can build a subspace spanned by k *low frequency basis vectors* out of the total N^2. For example if we want to keep only 25% of the data, we can build a $N^2/4$ dimensional subspace $\mathcal{T}$ that is spanned by $\{\boldsymbol{\varphi}_{m,n} \mid 0 \leq m, n \leq N/2-1\}$. In this case, the approximation

$$\hat{\mathbf{x}}_{25\%} = \sum_{m=0}^{N/2-1} \sum_{n=0}^{N/2-1} a_{m,n} \boldsymbol{\varphi}_{m,n} \tag{10}$$

is identified from the 2D array of coefficients:

$$\begin{matrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N/2-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N/2-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N/2-1,0} & a_{N/2-1,1} & \cdots & a_{N/2-1,N/2-1} \end{matrix}$$

These coefficients need to be found (similar to previous exercises) to build the optimal low dimensional approximation $\hat{\mathbf{x}}_{25\%}$.

Another common choice is to build a k dimensional subspace $\mathcal{T}$ using basis vectors $\boldsymbol{\varphi}_{m,n}$ with $m^2 + n^2 \leq r$ where $r > 0$ is appropriately picked resulting in k pair of non-negative integers m, n.

Experiment with an image (or several) and show a series of approximated images provided from various k-dimensional subspaces. Plot the error $\|\mathbf{x} - \hat{\mathbf{x}}_k\|$ versus k for some reasonable and illustrative set of k's.

This is, in principle, how JPEG compresses images or MPEG compresses movies. JPEG first divides the image into 8×8 blocks and applies a low dimensional approximation (according to the budget you specify) to each block. The coefficients $a_{m,n}$ are then quantized (e.g., rounded) and coded for storage.