CS663
Assignment 5 - Q5

Kalpesh Krishna
Pranav Sankhe
Mohit Madan

1 Which Error is Greater?

The correct statement is b), $E_L(\mathbf{V}) \geq E_N(\mathbf{V})$

2 Reason

For every vector $\mathbf{x}_i$, a non-linear approximation can equal to the linear approximation as long as there is no other $\mathbf{c}_i$ for which $||\mathbf{x}_i - \mathbf{V}\mathbf{c}_i||$ is lower. Hence, the non-linear approximation subsumes the linear approximation and must necessarily have a $\leq$ error.
The eigen-vectors give us the direction of variance across all data points taken together. A linear approximation constrains itself to just the top k eigen-vectors. However, it might be possible that certain $\mathbf{x}_i$ are out-liers in the dataset and are more correctly aligned with eigen-vectors not in the top k. Since a non-linear approximation draws a different set of k eigen-vectors for each $\mathbf{x}_i$, it approximates each $\mathbf{x}_i$ the best possible.

3 Algorithm

Since we are dealing with an orthonormal subspace of eigen-vectors, approximations are better with the most components. The algorithm is outlined below. Here the function topn() returns a vector with non-zero values only in the k largest values.
Time Complexity - The **for** loop will need $O(d^2)$ computations since each dot product is an $O(d)$ step. The topn() function essentially needs an order-k statistic, which can be solved in expected time $O(d)$ using quickselect. Hence the overall order is $O(d^2)$.
Correctness - This algorithm essentially checks how strongly correlated is $\mathbf{x}_i$ with each eigen-vector. Thereafter, the best k components are chosen. This is equivalent to the optimization problem outlined in the question. abs() is necessary since the negative of an eigen-vector is also a unit eigen-vector which can be a part of $\mathbf{V}$.

Algorithm

input : vector $\mathbf{x}_i$, eigen-vectors $\mathbf{V}(:, \mathbb{R}^d)$
output: best vector c_i
prods = zeros(size($\mathbf{V}$));
for $i \leftarrow 0$ **to** *size($\mathbf{V}$)* **do**
  prods(i) = abs($\mathbf{x}_i^T \cdot \mathbf{V}(i)$);
end
c_i = topn(prods, k);