Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 4

Student Name: Abhishek Kumar
Roll Number: 18111002
Date: November 8, 2018

QUESTION
1

Eigenchangers!

Given,
We want to compute eigen-vector (u) of X^TX given eigen-vector (v) of XX^T.

Note: Ignoring $\frac{1}{N}$ since eigen vectors will not change direction.

From property of eigen-vectors for a non zero eigen-value(λ) we can write:

$$XX^Tv = \lambda v$$

Multiplying X^T both sides we get:

$$(X^TX)(X^Tv) = \lambda(X^Tv)$$

Again, from eigen-vector property, we can say that X^Tv is an eigen-vector of X^TX for any non-zero eigen-value λ.

For zero eigen-value we will have:

$$X^TXv = 0 \implies v^TX^TXv = 0 \implies Xv = 0$$

So eigen-vectors for zero-eigen values can be obtained from null-space of X matrix in similar way to that of non-zero eigen value given we know v from null space.

Observation : Eigen-values of X^TX and XX^T will be same and are square of singular values we obtain in diagonal matrix of SVD decomposition.

Advantage of doing PCA this way :
For doing PCA we need to do $O(D^3)$ computation, but if $(D > N)$ then it is more computationally expensive as compared to $O(N^3 + ND)$ which is required in this method.

This concludes the answer to above question.

Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 4

Student Name: Abhishek Kumar
Roll Number: 18111002
Date: November 8, 2018

QUESTION

2

A General Activation Function

Given,

$$h(x) = \frac{x}{1 + \exp(-\beta x)}$$

1. Linear Function $\{\beta = 0\}$

When $\beta = 0$,

$$h(x) = \frac{x}{1 + \exp(0)} = \frac{x}{2}$$

So it becomes a linear function.

2. ReLU Function $\{\beta \to \infty\}$

When $\beta \to \infty$ then,

$$h(x) = \frac{x}{1 + \exp(-\infty x)} = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}$$

So it becomes ReLU activation function.

NOTE : There are different forms of this activation function depending on what we choose β to be. It can become reflections of ReLU along Y-axis as well and many other non-linear forms. That's why with introduction of one variable we have obtained a more general and non-linear activation function.

This concludes the answer to this question.

Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 4

Student Name: Abhishek Kumar
Roll Number: 18111002
Date: November 8, 2018

QUESTION
3

Mixtures meets Nueral Nets!

Given,

$$\{x_n, y_n\}_{n=1}^N,\ x_n \epsilon \mathbb{R}^D,\ y_n \epsilon \{0,1\}$$
$$z_n \sim multinoulli(\{\pi_k\}_{k=1}^K)$$
$$y_n \sim Bernoulli(\sigma(w_{z_n}^T x_n))$$

Now we can write out marginal probability as:

$$\begin{aligned} p(y_n = 1|x_n) &= \sum_{k=1}^K p(y_n = 1, z_n = k|x_n) \\ &= \sum_{k=1}^K p(y_n = 1|z_n = k, x_n) p(z_n = k|x_n) \\ &= \sum_{k=1}^K \pi_k \sigma(w_k^T x_n) \\ &= \sum_{k=1}^K \pi_k \sigma(\sum_{d=1}^D w_{kd} x_{nd}) = \sum_{k=1}^K \pi_k h_k \end{aligned}$$

Now this can be represented as a Nueral Network with a single hidden layer, W and π as weight matrices for first and second layer respectively, σ as activation function for hidden layer and x_n as input and $, p(y_n = 1|x_n)$ as output respectively. Output of hidden layer node is $h_k = \sigma(\sum_{d=1}^D w_{kd} x_{nd})$
See figure below:

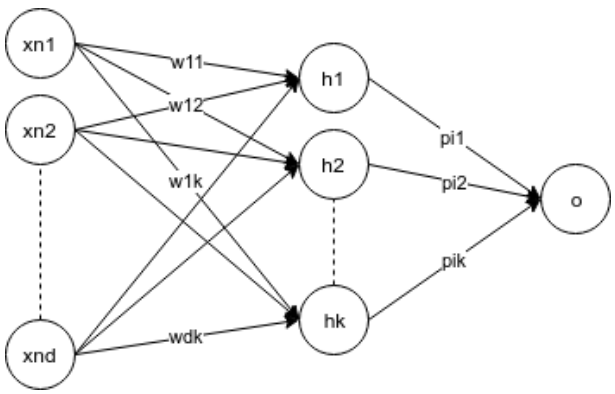

Figure 1: Equivalent Neural Network

This Concludes the answer to this problem.

Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 4

Student Name: Abhishek Kumar
Roll Number: 18111002
Date: November 8, 2018

QUESTION

4

Probabilistic Formulation of Matrix Factorization with Side Information

Given,
We have a matrix factorization problem over X which represents users and items as rows and columns respectively.

Model Assumptions

$$\begin{aligned} X_{nm} &\sim \mathcal{N}(X_{nm}|\theta_n + \phi_m + u_n^T v_m, \lambda_x^{-1}) \\ u_n &\sim \mathcal{N}(u_n|W_u a_n, \lambda_u^{-1} I_k) \\ v_m &\sim \mathcal{N}(v_m|W_v b_m, \lambda_v^{-1} I_k) \end{aligned}$$

where, $\{X_{nm}, \theta_n, \phi_m\} \epsilon \mathbb{R}$, $\{u_n, v_m\} \epsilon \mathbb{R}^K$, $W_u \epsilon \mathbb{R}^{K \times D_u}, W_v \epsilon \mathbb{R}^{K \times D_v}$, $a_n \epsilon R^{D_u}$, $b_m \epsilon R^{D_v}$
We want to estimate $\Theta = \{\{u_n, \theta_n\}_{n=1}^N, \{v_m, \phi_m\}_{m=1}^M, W_u, W_v\}$ given X where Ω is set of known indexes.

Loss Function

We want to do ALT-OPT optimization over this modelling so we can write is as maximization problem over know observation by deducing MAP as follows:

$$\begin{aligned} \hat{\Theta} &= \arg\max_{\Theta} p(X_\Omega, U, V) \\ &= \arg\max_{\Theta} p(X_\Omega|U, V) p(U, V) \\ &= \arg\max_{\Theta} p(X_\Omega|U, V) p(U) p(V) \\ &= \arg\max_{\Theta} [\log(p(X_\Omega|U, V)) + \log(p(U)) + \log(P(V))] \\ &= \arg\min_{\Theta} L(\Theta) \end{aligned}$$

Now we can write based on our assumptions .

$$\log(p(X_\Omega|U, V)) = C_0 - \sum_{\{n,m\} \epsilon \Omega} \frac{\lambda_x}{2} (X_{nm} - \theta_n - \phi_m - u_n^T v_m)^2$$

$$\log(p(U)) = C_1 - \sum_{n=1}^{N} \frac{\lambda_u}{2} (u_n - W_u a_n)^T (u_n - W_u a_n)$$

$$\log(p(V)) = C_2 - \sum_{m=1}^{M} \frac{\lambda_v}{2} (v_m - W_v b_m)^T (v_m - W_v b_m)$$

Now we can write our loss function or cost function ignoring constants as follows:

$$L(\Theta) = \sum_{\{n,m\}\epsilon\Omega} \frac{\lambda_x}{2}(X_{nm} - \theta_n - \phi_m - u_n^T v_m)^2 + \sum_{n=1}^{N} \frac{\lambda_u}{2}(u_n - W_u a_n)^T(u_n - W_u a_n)$$

$$+ \sum_{m=1}^{M} \frac{\lambda_v}{2}(v_m - W_v b_m)^T(v_m - W_v b_m)$$

ALT-OPT updates

Now we can take derivative equate to 0 to get update equations as follows:

$$\frac{\partial L}{\partial u_n} = (\sum_{m\epsilon\Omega_{r_n}} \lambda_x(X_{nm} - \theta_n - \phi_m - u_n^T v_m)(-v_m)) + \lambda_u(u_n - W_u a_n) = 0$$

$$\implies \boxed{u_n = \Big[\sum_{m\epsilon\Omega_{r_n}} \lambda_x v_m v_m^T + \lambda_u I_k\Big]^{-1}\Big[\sum_{m\epsilon\Omega_{r_n}} \lambda_x(X_{nm} - \theta_n - \phi_m)v_m + \lambda_u W_u a_n\Big]}$$

$$\frac{\partial L}{\partial \theta_n} = \sum_{m\epsilon\Omega_{r_n}} \lambda_x(X_{nm} - \theta_n - \phi_m - u_n^T v_m)(-1) = 0$$

$$\implies \boxed{\theta_n = \frac{1}{|\Omega_{r_n}|}\Big[\sum_{m\epsilon\Omega_{r_n}} (X_{nm} - \phi_m - u_n^T v_m)\Big]}$$

$$\frac{\partial L}{\partial W_u} = \sum_{n=1}^{N} \lambda_u(u_n - W_u a_n)(-a_n^T) = 0$$

$$\implies \boxed{W_u = \Big[\sum_{n=1}^{N} u_n a_n^T\Big]\Big[\sum_{n=1}^{N} a_n a_n^T\Big]^{-1}}$$

Similarly and symmetrically we get :

$$\boxed{\begin{aligned} v_m &= \Big[\sum_{n\epsilon\Omega_{c_m}} \lambda_x u_n u_n^T + \lambda_v I_k\Big]^{-1}\Big[\sum_{n\epsilon\Omega_{c_m}} \lambda_x(X_{nm} - \theta_n - \phi_m)u_n + \lambda_v W_v b_m\Big] \\ \phi_m &= \frac{1}{|\Omega_{c_m}|}\Big[\sum_{n\epsilon\Omega_{c_m}} (X_{nm} - \theta_n - u_n^T v_m)\Big] \\ W_v &= \Big[\sum_{m=1}^{M} v_m b_m^T\Big]\Big[\sum_{m=1}^{M} b_m b_m^T\Big]^{-1} \end{aligned}}$$

This concludes the answer to given problem.

Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 4

Student Name: Abhishek Kumar
Roll Number: 18111002
Date: November 8, 2018

QUESTION
5

Programming Problem 1

Given to apply PPCA with constant sigma using ALT-OPT method. which is nothing but PCA model itself without eigen-decomposition and orthogonality constraint over basis vectors.

ALT-OPT PPCA const. Sigma

Alternating optimization over PPCA with constan sigma basically boils down to two steps:

1. Update Z keeping W constant
$$\hat{Z} = XW[W^TW]^{-1}$$
2. Update W keeping Z constant
$$\hat{W} = XZ[Z^TZ]^{-1}$$

Doing this repeatedly for 200 iterations we will come to following solution:

Original Images were:

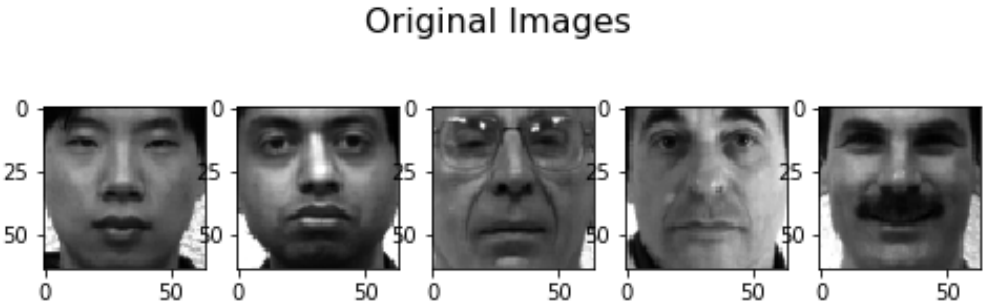

Figure 2: Original Images

Figure 3: Reconstructed Images

Basis Images (Columns of W)

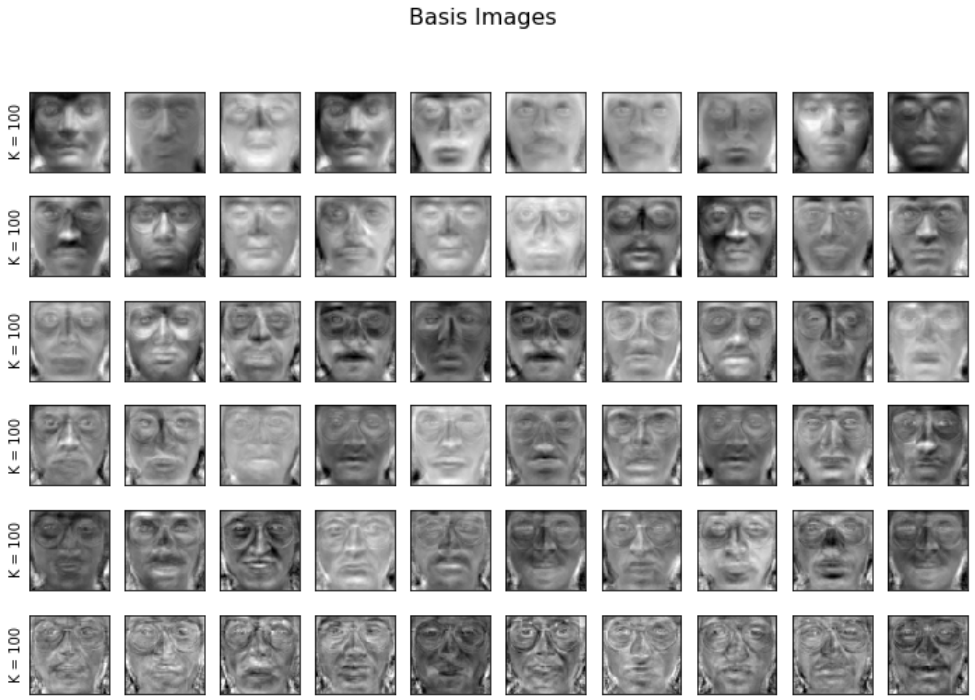

Figure 4: 10 Basis Images

Observation

The Reconstructed Images are worse than the original ones for low value of K but for large values they seem fine. Also face reconstructions are better with large K since it has more face parameters to take combinations.

Observation

The more i increase the number of basis size (K) the features that are learned are more like edges of faces as compared to low K size where it's mostly faces themselves which decreases the fluidity of models with less K and hence their accuracy.

Programming Problem 2

Given to implement PCA and TSNE with Kmeans Clustering over mnist dataset.

PCA Clustering

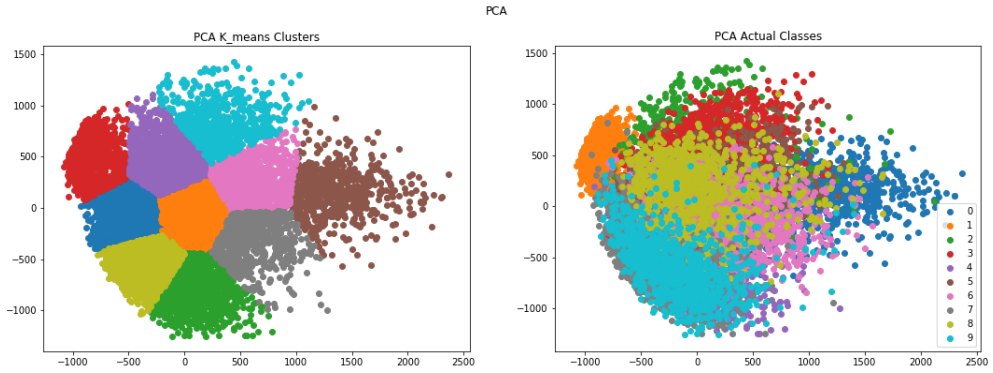

Figure 5: PCA Clustering vs Actual Classes

TSNE Clustering

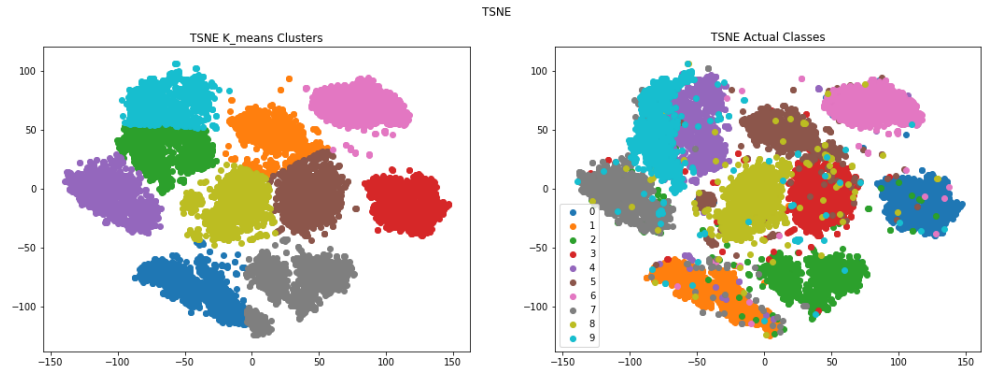

Figure 6: TSNE Clustering vs Actual Classes

Observation :

TSNE gives a much better clustering as compared to PCA for classification purpose since PCA simply looks at variance whereas TSNE looks at neighbourhood and maintains the relation ship.

Some random runs over kmeans after PCA and TSNE reduction.

Figure 7: TSNE vs PCA Clusterings

We can observed how PCA clusters are merged and TSNE clusters are some what far apart and so we can conclude that TSNE gives better results visually as compared to PCA.
This concludes the answer to this question.