CS 189: Introduction to Machine Learning

Fall 2017

Homework 5

Solutions by

Jinhong Du

3033483677

Question 1

(a)

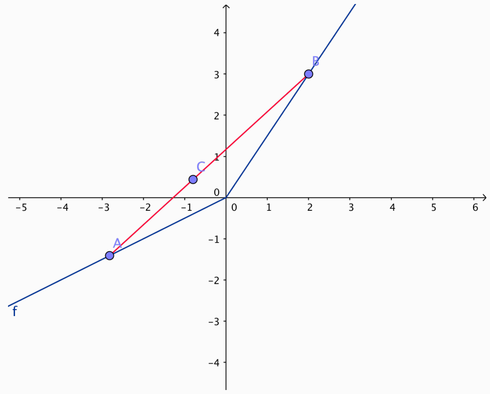
Jinhong Du
jaydu@berkeley.edu

(b)

I certify that all solutions are entirely in my words and that I have not looked at another student's solutions. I have credited all external sources in this write up. Jinhong Du

Question 2

(a)



$$f(x) = \begin{cases} x & , x \geqslant 0 \\ \frac{1}{2}x & , x < 0 \end{cases}$$

Obiviously, $f : \mathbb{R} \to \mathbb{R}$, and $f(x)$ is not differentiable at 0 since $f'(0+) = 1 \neq 2 = f'(0-)$. And $f(x)$ is a convex function.

(b)

$$\begin{aligned} f(x) &= \|Ax - y\|^2 \\ &= (Ax - y)^T(Ax - y) \\ &= (x^TA^T - y^T)(Ax - y) \\ &= x^TA^TAx - x^TA^Ty - y^TAx + y^Ty \\ \Delta_x f &= \frac{\partial f}{\partial x} \\ &= 2A^TAx - 2A^Ty \\ H &= \frac{\partial f^2}{\partial x \partial x^T} \\ &= 2A^TA \end{aligned}$$

Suppose that the SVD of A is $A = U\Sigma V^T$,

$\because \quad \forall\ v \in \mathbb{R}^d$, $V^Tv = v' \in \mathbb{R}^d$

$$\begin{aligned} v^TA^TAv &= v^T(U\Sigma V^T)^T(U\Sigma V^T)v \\ &= v^TV\Sigma U^TU\Sigma V^Tv \\ &= (V^Tv)^T\Sigma^2(V^Tv) \end{aligned}$$

and $\Sigma^2 = \begin{bmatrix} \sigma_1^2 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_d^2 \end{bmatrix}$ is a diagonal matrix with non-negetive diagonal entries

Solution (cont.)

$\therefore$

$$v^T A^T A v = \sum_{i=1}^{d} \sigma_i^2 v_i'^2 \geqslant 0$$

i.e. the Hessian of f is positive semi-definite.

(c)

Suppose that both x_1, x_2 in the domain of f are local minimums of $f(x)$.
If $f(x_1) > f(x_2)$, $\forall\ n \in \mathbb{N}^+$,

$$f(x_1) > \left(1 - \frac{1}{n}\right) f(x_1) + \frac{1}{n} f(x_2) \geqslant f\left(\left(1 - \frac{1}{n}\right) x_1 + \frac{1}{n} x_2\right)$$

Let $y_n = \left(1 - \frac{1}{n}\right) x_1 + \frac{1}{n} x_2$, then $\lim_{n\to+\infty} y_n = x_1$ and $f(x_1) > f(y_n)$
Therefore, $\forall\ \epsilon > 0$, $\exists\ n \in \mathbb{N}^+$, s.t. $\|x_1 - y_n\| < \delta$ and $f(x_1) > f(y_n)$
There is a contradiction. Therefore, $f(x_1) \geqslant f(x_2)$. Similarly, we have $f(x_1) \geqslant f(x_2)$
Therefore, $f(x_1) = f(x_2)$, i.e. if $f(x)$ has more than one local minimun, then the values of these local minimum should be the same.
If x_0 is a local minimun of f and not the global minimum, then $\exists\ y$ in the domain of $f(x)$, s.t. $f(y) < f(x_0)$, $\forall\ n \in \mathbb{N}^+$,

$$f(y) > \left(1 - \frac{1}{n}\right) f(x_0) + \frac{1}{n} f(y) \geqslant f\left(\left(1 - \frac{1}{n}\right) x_0 + \frac{1}{n} y\right)$$

Let $z_n = \left(1 - \frac{1}{n}\right) x_0 + \frac{1}{n} y$, then $\lim_{n\to+\infty} z_n = x_0$ and $f(y) > f(z_n)$
$\because$ x_0 is the local minimum of $f(x)$
$\therefore$ $\forall\ \epsilon > 0$, $\exists\ n \in \mathbb{N}^+$, s.t. $\|x_0 - z_n\| < \delta$ and $f(x_0) < f(z_n) < f(y)$
There is a contradiction. Therefore, $f(x_0)$ is the global minimum.

(d)

$\because$ $f(x), g(x)$ are convex function
$\therefore$ $\forall\ \lambda \in [0, 1]$, $\forall\ x_1, x_2 \in S_f$,

$$\lambda f(x_1) + (1 - \lambda) f(x_2) \geqslant f(\lambda x_1 + (1 - \lambda) x_2)$$

$$\lambda g(x_1) + (1 - \lambda) g(x_2) \geqslant g(\lambda x_1 + (1 - \lambda) x_2)$$

(i)
$\because$

$$\lambda[f(x_1) + g(x_1)] + (1 - \lambda)[f(x_2) + g(x_2)] \geqslant f(\lambda x_1 + (1 - \lambda) x_2) + g(\lambda x_1 + (1 - \lambda) x_2)$$

$\therefore$

$$\lambda h(x_1) + (1 - \lambda) h(x_2) \geqslant h(\lambda x_1 + (1 - \lambda) x_2)$$

i.e. $h(x)$ is convex
(ii)

$$f(x) = x$$

Solution (cont.)

$$g(x) = 2x$$

$\because$

$$\lambda f(x_1) + (1-\lambda) f(x_2) = \lambda x_1 + (1-\lambda)x_2 = f(\lambda x_1 + (1-\lambda)x_2)$$

$$\lambda g(x_1) + (1-\lambda) g(x_2) = 2\lambda x_1 + 2(1-\lambda)x_2 = g(\lambda x_1 + (1-\lambda)x_2)$$

$\therefore \quad f(x), g(x)$ are convex

However, $h(x) = \begin{cases} x & , x \geqslant \\ 2x & , x < 0 \end{cases}$ and $x_1 = -1, x_2 = 1, \lambda = \frac{1}{2}$,

$$\begin{aligned} \lambda h(x_1) + (1-\lambda) h(x_2) &= \frac{1}{2} g(-1) + \frac{1}{2} f(1) \\ &= -\frac{1}{2} \cdot 2 + \frac{1}{2} \\ &= -\frac{1}{2} \\ &< 0 = h(-1 \cdot \frac{1}{2} + 1 \cdot \frac{1}{2}) \end{aligned}$$

i.e. $h(x)$ is not convex.

(iii)

$\because$

$$\begin{aligned} \lambda \max\{f(x_1), g(x_1)\} + (1-\lambda) \max\{f(x_2) + g(x_2)\} &\geqslant \lambda f(x_1) + (1-\lambda) f(x_2) \\ &\geqslant f(\lambda x_1 + (1-\lambda)x_2) \\ \lambda \max\{f(x_1), g(x_1)\} + (1-\lambda) \max\{f(x_2) + g(x_2)\} &\geqslant \lambda g(x_1) + (1-\lambda) g(x_2) \\ &\geqslant g(\lambda x_1 + (1-\lambda)x_2) \end{aligned}$$

$\therefore$

$$\lambda \max\{f(x_1), g(x_1)\} + (1-\lambda) \max\{f(x_2) + g(x_2)\} \geqslant \max\{f(\lambda x_1 + (1-\lambda)x_2), g(\lambda x_1 + (1-\lambda)x_2)\}$$

$\therefore$

$$\lambda h(x_1) + (1-\lambda) h(x_2) \geqslant h(\lambda x_1 + (1-\lambda)x_2)$$

i.e. $h(x)$ is convex

(iv)

$$f(x) = -x$$

$$g(x) = x^2$$

$\because$

$$\lambda f(x_1) + (1-\lambda) f(x_2) = -\lambda x_1 - (1-\lambda)x_2 = f(\lambda x_1 + (1-\lambda)x_2)$$

$$g^{(2)}(x) = 2 > 0$$

$\therefore \quad f(x), g(x)$ are convex

However, $h(x) = f(g(x)) = -x^2$ and $h^{(2)}(x) = -2 < 0$, i.e. $h(x)$ is not convex.

Question 3

(a)

$$
\begin{aligned}
A &= U\Sigma V^T \\
&= \begin{bmatrix} u_1 & u_2 & \cdots & u_n \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_d \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \cdots \\ v_d^T \end{bmatrix} \\
&= \begin{bmatrix} u_1\sigma_1 & u_2\sigma_2 & \cdots & u_d\sigma_d \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \cdots \\ v_d^T \end{bmatrix} \\
&= \sum_{i=1}^{d} u_i \sigma_i v_i^T \\
&= \sum_{i=1}^{d} \sigma_i u_i v_i^T
\end{aligned}
$$

(b)

$\because$ Σ is a diagonal matrix, U, V have orthonormal columns

$\therefore$ $U^T T = V^T V = I$

$$
\begin{aligned}
\Sigma^T\Sigma &= \begin{bmatrix} \sigma_1 & 0 & 0 & \cdots & 0 & 0\cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 & 0\cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & 0\cdots & 0 \\ 0 & 0 & 0 & \cdots & \sigma_d & 0\cdots & 0 \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_d \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \\
&= \begin{bmatrix} \sigma_1^2 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_d^2 \end{bmatrix}
\end{aligned}
$$

Solution (cont.)

$\therefore$

$$\begin{aligned} A^TA &= (U\Sigma V^T)^T(U\Sigma V^T) \\ &= V\Sigma^T U^T U\Sigma V^T \\ &= V\Sigma^T\Sigma V^T \end{aligned}$$

$\because$

$$\begin{aligned} A^TAv_i &= V\Sigma^T\Sigma V^T v_i \\ &= \begin{bmatrix} v_1 & v_2 & \cdots & v_d \end{bmatrix} \begin{bmatrix} \sigma_1^2 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_d^2 \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \cdots \\ v_d^T \end{bmatrix} v_i \\ &= \begin{bmatrix} v_1 & v_2 & \cdots & v_d \end{bmatrix} \begin{bmatrix} \sigma_1^2 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_d^2 \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \sigma_i^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ &= \sigma_i^2 v_i \end{aligned}$$

$\therefore$ A^TA has ith eigenvalue $\lambda_i = \sigma_i^2$ with associated eigenvector v_i $(1 \leqslant i \leqslant d)$..

$\because$

$$\begin{aligned} AA^T &= (U\Sigma V^T)(U\Sigma V^T)^T \\ &= U\Sigma V^T V\Sigma^T U^T \\ &= U\Sigma\Sigma^T U^T \end{aligned}$$

Solution (cont.)

$$
\begin{aligned}
\Sigma\Sigma^T &= \begin{bmatrix} \sigma_1 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_d \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 & \cdots & 0 & 0 \cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 & 0 \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & 0 \cdots & 0 \\ 0 & 0 & 0 & \cdots & \sigma_d & 0 \cdots & 0 \end{bmatrix} \\
&= \begin{bmatrix} \sigma_1^2 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & \sigma_d^2 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 \end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
AA^T u_i &= U\Sigma\Sigma^T U^T u_i \\
&= \begin{bmatrix} u_1 & u_2 & \cdots & u_n \end{bmatrix} \begin{bmatrix} \sigma_1^2 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & \sigma_d^2 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} u_1^T \\ u_2^T \\ \cdots \\ u_n^T \end{bmatrix} u_i \\
&= \begin{bmatrix} u_1 & u_2 & \cdots & u_n \end{bmatrix} \begin{bmatrix} \sigma_1^2 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & \sigma_d^2 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} u_1 & u_2 & \cdots & u_n \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \sigma_i^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\
&= \sigma_i^2 u_i
\end{aligned}
$$

$\therefore$ AA^T has ith eigenvalue $\lambda_i = \sigma_i^2$ with associated eigenvector u_i $(1 \leqslant i \leqslant d)$.

(c)

Given $u \in \mathbb{R}^n, v \in \mathbb{R}^d, \|u\| = \|v\| = 1$, we have $u = \sum\limits_{i=1}^{n} a_i u_i, v = \sum\limits_{i=1}^{d} b_i v_i$, where $\sum\limits_{i=1}^{n} a_i = \sum\limits_{i=1}^{d} b_i = 1$, since $\{u_1, u_2, \cdots, u_n\}$ is the orthonormal basis of $\mathbb{R}^n$ and $\{v_1, v_2, \cdots, v_d\}$ is the orthronormal basis of $\mathbb{R}^d$. Therefore,

$$
\begin{aligned}
u^T A v &= \left(\sum_{i=1}^{n} a_i u_i\right)^T A \left(\sum_{i=1}^{d} b_i v_i\right) \\
&= \left(\sum_{i=1}^{n} a_i u_i^T\right) A \left(\sum_{i=1}^{d} b_i v_i\right) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{d} a_i b_j u_i^T U \Sigma V^T v_j \\
&= \sum_{i=1}^{n} \sum_{j=1}^{d} a_i b_j \begin{array}{c} \begin{array}{ccccccc} & & & i & & & \end{array} \\ \begin{pmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{pmatrix} \end{array} \Sigma \begin{array}{c} \\ \\ \\ j \\ \\ \\ \\ \end{array} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \\
&= \sum_{i=1}^{d} a_i b_i \sigma_i
\end{aligned}
$$

$\because \quad \sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_d$, $|a_i| \leqslant 1, \left|\sum\limits_{i=1}^{d} a_i b_i\right| \leqslant \max\limits_{i}\{|a_i|\} \left|\sum\limits_{i=1}^{d} b_i\right| \leqslant 1$

$\therefore$

$$
u^T A v \leqslant \sum_{i=1}^{d} a_i b_i \sigma_1 \leqslant \sigma_1
$$

and when $a_1 = b_1 = 1, a_2 = \cdots = a_n = b_2 = \cdots = b_d = 0$,

$$
u_1^T A v_1 = \sigma_1
$$

i.e. if A has a unique maximum singular value σ_1, then the maximizers (u^*, v^*) above are given by the first left and right singular vectors u_1, v_1, respectively.
Therefore,

$$
\sigma_1(A) = \max_{\substack{u:\|u\|=1 \\ v:\|v\|=1}} u^T A v
$$

(d)

$\because \quad a^T X, b^T Y \in \mathbb{R}$

$\therefore \quad a^T X = X^T a, b^T Y = Y^T b, (a^T X)^2 = (a^T X)(a^T X)^T = a^T X^T X a, (a^T X)^2 = (a^T X)(b^T Y)^T = b^T Y^T Y b,$

Solution (cont.)

$\therefore$

$$\begin{aligned}
\rho &= \max_{a,b\in\mathbb{R}^d} \rho(a^TX, b^TY) \\
&= \max_{a,b\in\mathbb{R}^d} \frac{\mathbb{E}(a^TXY^Tb)}{\sqrt{\mathbb{E}(a^TX)^2\mathbb{E}(Y^Tb)^2}} \\
&= \max_{a,b\in\mathbb{R}^d} \frac{a^T\mathbb{E}(XY^T)b}{\sqrt{\mathbb{E}(a^TX^TXa)\mathbb{E}(b^TY^TYb)}} \\
&= \max_{a,b\in\mathbb{R}^d} \frac{a^T\mathbb{E}(XY^T)b}{\sqrt{a^T\mathbb{E}(X^TX)ab^T\mathbb{E}(Y^TY)b}} \\
&= \max_{a,b\in\mathbb{R}^d} \frac{a^T\Sigma_{XY}b}{\sqrt{a^T\Sigma_{XX}ab^T\Sigma_{YY}b}} \\
&= \max_{a,b\in\mathbb{R}^d} \frac{a^T\Sigma_{XY}b}{(a^T\Sigma_{XX}a)^{\frac{1}{2}}(b^T\Sigma_{YY}b)^{\frac{1}{2}}}
\end{aligned}$$

If (a^*, b^*) is a maximizer above, then $\forall\ \alpha, \beta > 0$,

$$\begin{aligned}
\rho' &= \frac{(\alpha a^*)^T\Sigma_{XY}(\beta b^*)}{[(\alpha a^*)^T\Sigma_{XX}(\alpha a^*)]^{\frac{1}{2}}[(\beta b^*)^T\Sigma_{YY}(\beta b^*)]^{\frac{1}{2}}} \\
&= \frac{\alpha\beta a^{*T}\Sigma_{XY}b^*}{\alpha\beta(a^{*T}\Sigma_{XX}a^*)^{\frac{1}{2}}(b^{*T}\Sigma_{YY}b^*)^{\frac{1}{2}}} \\
&= \frac{a^{*T}\Sigma_{XY}b^*}{(a^{*T}\Sigma_{XX}a^*)^{\frac{1}{2}}(b^{*T}\Sigma_{YY}b^*)^{\frac{1}{2}}} \\
&= \rho
\end{aligned}$$

Therefore, $(\alpha a^*, \beta b^*)$ is also a maximizer for any $\alpha, \beta > 0$

(e)

i)

From (d), we know ρ is scaling invariant. Let $c = \frac{\Sigma_{XX}^{\frac{1}{2}}a}{\|\Sigma_{XX}^{\frac{1}{2}}a\|}$, $d = \frac{\Sigma_{YY}^{\frac{1}{2}}b}{\|\Sigma_{YY}^{\frac{1}{2}}b\|}$, then

$$\tilde{\rho} = \frac{c^T\Sigma_{XX}^{-\frac{1}{2}}\Sigma_{XY}\Sigma_{YY}^{-\frac{1}{2}}d}{\sqrt{c^Tc}\sqrt{d^Td}}$$

By the Cauchy-Schwarz inequality,

$$\begin{aligned}
\left(c^T\Sigma_{XX}^{-\frac{1}{2}}\Sigma_{XY}\Sigma_{YY}^{-\frac{1}{2}}\right)d &\leqslant \left(c^T\Sigma_{XX}^{-\frac{1}{2}}\Sigma_{XY}\Sigma_{YY}^{-\frac{1}{2}}\Sigma_{YY}^{-\frac{1}{2}}\Sigma_{YX}\Sigma_{XX}^{-\frac{1}{2}}c\right)^{\frac{1}{2}}(d^Td)^{\frac{1}{2}} \\
&= \left(c^T\Sigma_{XX}^{-\frac{1}{2}}\Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX}\Sigma_{XX}^{-\frac{1}{2}}c\right)^{\frac{1}{2}}(d^Td)^{\frac{1}{2}}
\end{aligned}$$

equality holds when d and $\Sigma_{YY}^{-\frac{1}{2}}\Sigma_{YX}\Sigma_{XX}^{-\frac{1}{2}}c$ are colinear.

Solution (cont.)

Therefore

$$\begin{aligned}\tilde{\rho} &\leqslant \frac{\left(c^T\Sigma_{XX}^{-\frac{1}{2}}\Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX}\Sigma_{XX}^{-\frac{1}{2}}c\right)^{\frac{1}{2}}}{(c^Tc)^{\frac{1}{2}}} \\ &= \left(\frac{c^T\Sigma_{XX}^{-\frac{1}{2}}\Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX}\Sigma_{XX}^{-\frac{1}{2}}c}{c^Tc}\right)^{\frac{1}{2}}\end{aligned}$$

Denote $M = \Sigma_{XX}^{-\frac{1}{2}}\Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX}\Sigma_{XX}^{-\frac{1}{2}}$, $M^T = M$, $R(M,x) = \dfrac{x^TMx}{x^Tx}$. By the Lagrange multipliers, to find the critical points of $R(M,x)$, s.t. $\|x\| = 1$, is the same to find the critical points of

$$L(x) = x^TMx - \lambda(x^Tx - 1)$$

Let

$$\frac{\mathrm{d}L}{\mathrm{d}x} = 2Mx - 2\lambda x = 0$$

We have

$$Mx = \lambda x$$

and

$$R(M,x) = \frac{x^TMx}{x^Tx} = \lambda$$

i.e. the eigenvectors $x_1, \cdots, x_n$ of M are the critical points of the Rayleigh quotient and their corresponding eigenvalues $\lambda_1 > \cdots > \lambda_n$ are the stationary values of R.
Therefore, when $c = x_1$, d and $\Sigma_{YY}^{-\frac{1}{2}}\Sigma_{YX}\Sigma_{XX}^{-\frac{1}{2}}c$ are colinear, $\tilde{\rho}^2$ has global maximum λ_1, i.e. $\rho^2 = \lambda_1$.
ii)
$\because$

$$\rho = \frac{a^{*T}\Sigma_{XY}b^*}{(a^{*T}\Sigma_{XX}a^*)^{\frac{1}{2}}(b^{*T}\Sigma_{YY}b^*)^{\frac{1}{2}}}$$

$\therefore$ from i) we have

$$\frac{\Sigma_{XX}^{-\frac{1}{2}}a^*}{\|\Sigma_{XX}^{-\frac{1}{2}}a^*\|} = x_1$$

By the scale invariance of ρ, we can as well assume that $\Sigma_{XX}^{-\frac{1}{2}}a^* = x_1$.
iii)
Similarly to part i),

$$\begin{aligned}\left(d^T\Sigma_{YY}^{-\frac{1}{2}}\Sigma_{YX}\Sigma_{XX}^{-\frac{1}{2}}\right)c &\leqslant \left(d^T\Sigma_{YY}^{-\frac{1}{2}}\Sigma_{YX}\Sigma_{XX}^{-\frac{1}{2}}\Sigma_{XX}^{-\frac{1}{2}}\Sigma_{XY}\Sigma_{YY}^{-\frac{1}{2}}c\right)^{\frac{1}{2}}(c^Tc)^{\frac{1}{2}} \\ &= \left(d^T\Sigma_{YY}^{-\frac{1}{2}}\Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY}\Sigma_{YY}^{-\frac{1}{2}}c\right)^{\frac{1}{2}}(c^Tc)^{\frac{1}{2}}\end{aligned}$$

Then

$$\begin{aligned}\tilde{\rho} &= \frac{c^T\Sigma_{XX}^{-\frac{1}{2}}\Sigma_{XY}\Sigma_{YY}^{-\frac{1}{2}}d}{\sqrt{c^Tc}\sqrt{d^Td}} \\ &= \frac{d^T\Sigma_{YY}^{-\frac{1}{2}}\Sigma_{YX}\Sigma_{XX}^{-\frac{1}{2}}c}{\sqrt{c^Tc}\sqrt{d^Td}} \\ &\leqslant \left(\frac{d^T\Sigma_{YY}^{-\frac{1}{2}}\Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY}\Sigma_{YY}^{-\frac{1}{2}}d}{d^Td}\right)^{\frac{1}{2}}\end{aligned}$$

Solution (cont.)

Repeat the above steps, we can get that $\tilde{\rho}$ is maximized when $d = x_1$, c and $\Sigma_{XX}^{-\frac{1}{2}}\Sigma_{XY}\Sigma_{YY}^{-\frac{1}{2}}d$ are colinear. Thus,

$$\frac{\Sigma_{YY}^{-\frac{1}{2}}b^*}{\|\Sigma_{YY}^{-\frac{1}{2}}b^*\|} = y_1$$

where $y_1, \cdots, y_n$ are the eigenvectors of $\Sigma_{YY}^{-\frac{1}{2}}\Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY}\Sigma_{YY}^{-\frac{1}{2}}$, with eigenvalues $\lambda_1' > \cdots > \lambda_n'$ respectively.

By the scale invariance of ρ, we can as well assume that $\Sigma_{YY}^{-\frac{1}{2}}b^* = x_1$.

(f)

We know that when $cov(X_i, Y_j) = 0 \quad (\forall\ i, j \in \mathbb{N}^+, i, j \leqslant d)$, $\Sigma_{XY} = cov(X, Y) = 0$ and $\rho \equiv 0$. So the previous process to get a optimal solution for maximizing ρ will false. In other words, CCA is useless when the random vectors X and Y are uncorrelated.

Let $Y' = Y^2$, and apply CCA to X and Y' to see if ρ is close to ± 1. It is because that when $|\rho|$ is close to 1, a^TX and b^TY' will be more linear correlated, i.e. $a^TX \approx cb^TY'$, i.e. X and Y^2 share a linear relationship.

And if we know that $X = AY^2 + B$ where $A, B \in \mathbb{R}$, then

$$\begin{aligned}
\mathbb{E}(XY^T) &= \mathbb{E}[(AY^2 + B)Y^T] \\
&= A\mathbb{E}(Y^2Y^T) + B\mathbb{E}(Ones_{n\times 1}Y^T) \\
\mathbb{E}(XX^T) &= \mathbb{E}[(AY^2 + B)(AY^2 + B)^T] \\
&= A^2\mathbb{E}(Y^2Y^{2T}) + AB\mathbb{E}(Ones_{n\times 1}Y^{2T}) + AB\mathbb{E}(Y^2Ones_{1\times n}) + B^2
\end{aligned}$$

And calculate ρ.

Question 4

(a)

First, we rewite $\hat{X} = \begin{pmatrix} X_1 & X_2 & \cdots & X_n \end{pmatrix}^T$ and $\hat{Y} = \begin{pmatrix} Y_1 & Y_2 & \cdots & Y_n \end{pmatrix}^T$ where $X_i, Y_i \in \mathbb{R}^d$, in the training set: $n = 956$ and $d = 675$. Define the sample covariance matrix as

$$\begin{aligned}\hat{\Sigma}_{XY} &= \frac{1}{n-1}\begin{bmatrix} cov(X_1,Y_1) & cov(X_1,Y_2) & \cdots & cov(X_1,Y_n) \\ cov(X_2,Y_1) & cov(X_2,Y_2) & \cdots & cov(X_2,Y_n) \\ \vdots & \vdots & \vdots & \vdots \\ cov(X_n,Y_1) & cov(X_n,Y_2) & \cdots & cov(X_n,Y_n) \end{bmatrix} \\ &= \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})^T\end{aligned}$$

$\because$ suppose that X_i, Y_i(i.i.d) and $X, Y \in \mathbb{R}^d$ have same p.d.f respectively, then

$$\begin{aligned}\mathbb{E}\hat{\Sigma}_{XY} &= \frac{1}{n-1}\mathbb{E}\left[\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})^T\right] \\ &= \frac{1}{n-1}\mathbb{E}\left[\sum_{i=1}^{n}(X_iY_i^T - \overline{X}Y_i^T - X_i\overline{Y}^T + \overline{X}\overline{Y}^T)\right] \\ &= \frac{1}{n-1}\mathbb{E}\left(\sum_{i=1}^{n}X_iY_i^T - n\overline{X}\overline{Y}^T - n\overline{X}\overline{Y}^T + n\overline{X}\overline{Y}^T\right) \\ &= \frac{1}{n-1}\mathbb{E}\left(\sum_{i=1}^{n}X_iY_i^T - n\overline{X}\overline{Y}^T\right) \\ &= \frac{1}{n-1}\mathbb{E}\left(\sum_{i=1}^{n}X_iY_i^T - \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}X_iY_j^T\right) \\ &= \frac{1}{n-1}\mathbb{E}\left(\sum_{i=1}^{n}X_iY_i^T\right) - \frac{1}{n-1}\mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n}X_iY_i^T\right) \\ &= \frac{1}{n-1}\left[n\mathbb{E}(XY^T) - \mathbb{E}(XY^T)\right] \\ &= \mathbb{E}(XY^T)\end{aligned}$$

$\therefore$ $\hat{\Sigma}_{XY}$ is the unbaised estimator of Σ_{XY}.
Similarly, $\hat{\Sigma}_{XX}, \hat{\Sigma}_{YY}$ is the unbaised estimator of Σ_{XX}, Σ_{YY} respectively.
$\therefore$

$$\begin{aligned}\hat{\Sigma}_{XY} &= \frac{1}{n-1}X_zY_z^T \\ \hat{\Sigma}_{XX} &= \frac{1}{n-1}X_zX_z^T \\ \hat{\Sigma}_{YY} &= \frac{1}{n-1}Y_zY_z^T\end{aligned}$$

where $X_z = \begin{pmatrix} X_1 - \overline{X} & X_2 - \overline{X} & \cdots & X_n - \overline{X} \end{pmatrix}^T, Y_z = \begin{pmatrix} Y_1 - \overline{Y} & Y_2 - \overline{Y} & \cdots & Y_n - \overline{Y} \end{pmatrix}^T$.

```
import pickle
```

Solution (cont.)

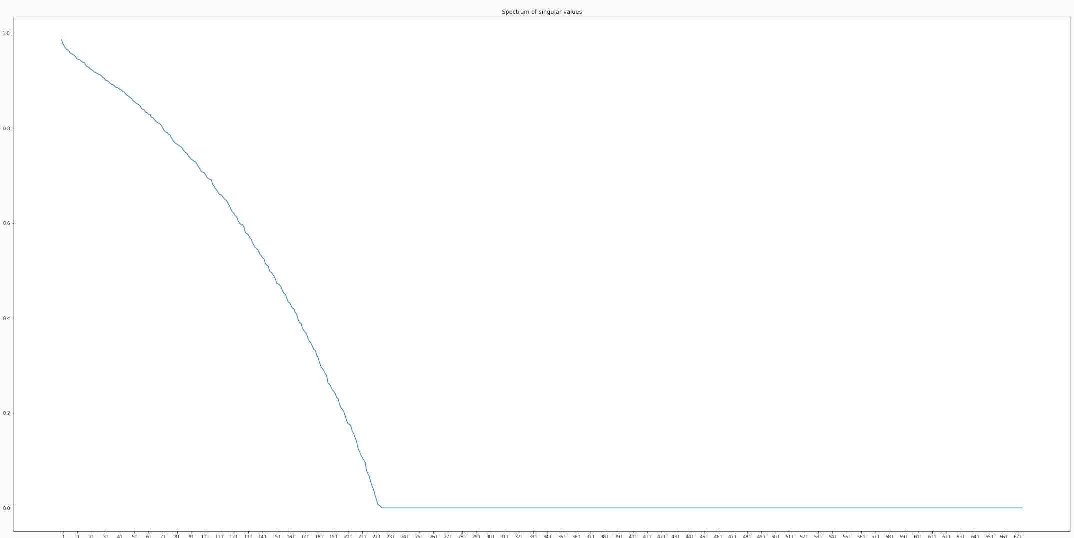
```
import numpy as np
from scipy.linalg import eig
from scipy.linalg import sqrtm
from numpy.linalg import inv
from numpy.linalg import svd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
with open('hw05-data/x_train.p','rb') as f:
    s = f.read()
    s = s.replace(b'\r',b'')
with open('hw05-data/x_train.p','wb') as f:
    f.write(s)
with open('hw05-data/y_train.p','rb') as f:
    s = f.read()
    s = s.replace(b'\r',b'')
with open('hw05-data/y_train.p','wb') as f:
    f.write(s)
with open('hw05-data/x_test.p','rb') as f:
    s = f.read()
    s = s.replace(b'\r',b'')
with open('hw05-data/x_test.p','wb') as f:
    f.write(s)
with open('hw05-data/y_test.p','rb') as f:
    s = f.read()
    s = s.replace(b'\r',b'')
with open('hw05-data/y_test.p','wb') as f:
    f.write(s)
with open('hw05-data/x_train.p','rb') as f:
    x_train = pickle.load(f,encoding='latin1')
with open('hw05-data/y_train.p','rb') as f:
    y_train = pickle.load(f,encoding='latin1')
x_train = np.array(x_train)
y_train = np.array(y_train)
Xtrain = np.reshape(x_train,(len(x_train),
         np.prod(np.shape(x_train)[1:])),order='F')
Ytrain = np.reshape(y_train,(len(y_train),
        np.prod(np.shape(y_train)[1:])),order='F')
Xtrain = Xtrain/255*2.0-1.0
Xtrain = Xtrain.T
Ytrain = Ytrain/255*2.0-1.0
Ytrain = Ytrain.T
n = len(Xtrain)
xmean = np.mean(Xtrain,axis=1)
```

Solution (cont.)

```
ymean = np.mean(Ytrain, axis=1)
sigmaxy = (Xtrain-xmean[:,np.newaxis]).dot(
            (Ytrain-ymean[:,np.newaxis]).T)/(n-1)
sigmaxx = (Xtrain-xmean[:,np.newaxis]).dot(
            (Xtrain-xmean[:,np.newaxis]).T)/(n-1)
sigmayy = (Ytrain-ymean[:,np.newaxis]).dot(
            (Ytrain-ymean[:,np.newaxis]).T)/(n-1)
```

(b)

We can see that the sigular values decay as the order increases.

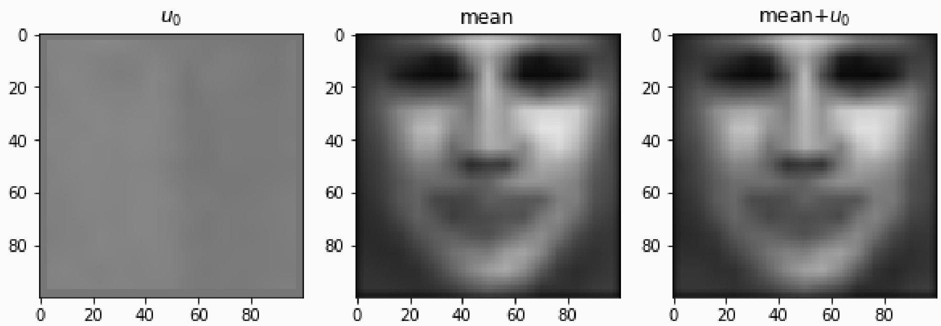



```
lamb = 0.00001
M = inv(sqrtm(sigmaxx+lamb)).dot(sigmaxy).dot(inv(sqrtm(sigmayy+lamb)))
[u, s, vt] = svd(M)
plt.figure(figsize=(40,20))
plt.plot(s)
plt.xticks(np.arange(1,m+1,10))
plt.title('Spectrum of singular values')
plt.show()
print(s)
```

(c)

We can see that projecting the training data to subspace $\{u_0\}$ is not good enough to represent the main feature of face.

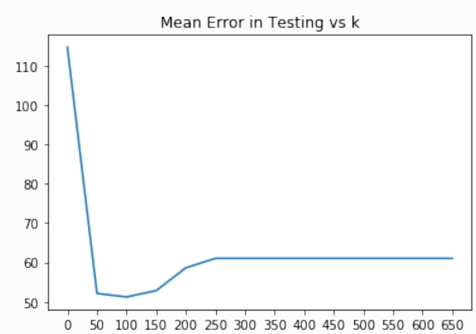
Solution (cont.)




```python
from skimage.transform import resize
from skimage.io import imsave

def plot_image(vector):
  vector = ((vector+1.0) / 2.0) * 255.0
  vector = np.reshape(vector, (15, 15, 3),order='F')
  p = vector.astype("uint8")
  p = resize(p, (100, 100))
  plt.imshow(p)
  imsave('eigenface.png',p)
```

(d)

We can see that the testing error goes down first and goes up later. It is minimized when $k = 100$.




```python
def regreession_multi(X,y,xtest,ytest,lamb):
  if len(np.shape(X))<2:
      X= X[:,np.newaxis]
  if len(np.shape(xtest))<2:
      xtest= xtest[:,np.newaxis]
  n1, n2 = np.shape(X)
  A = np.linalg.solve(X.T.dot(X)+lamb*np.identity(n2),X.T.dot(y))
  yhat = X.dot(A)
```

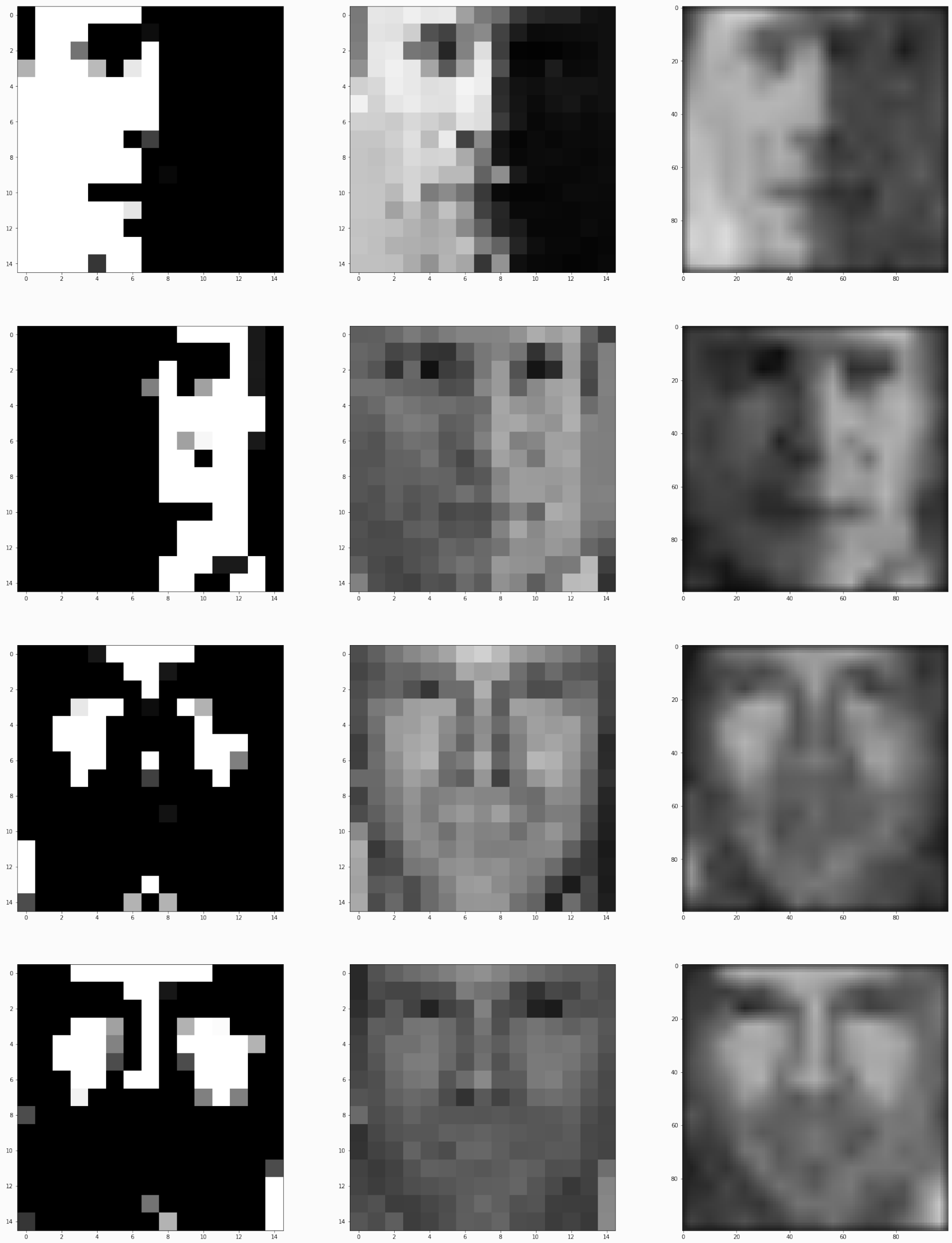
Solution (cont.)

```

    Rmean =  np.linalg.norm(yhat-y)**2/n1

    yhat_test = xtest.dot(A)
    Rmean_test = np.linalg.norm(yhat_test-ytest)**2/len(ytest)

    return {'A':A,'train_error':Rmean,'test_error':Rmean_test}
k = np.arange(0,700,50)
result= []
for i in range(len(k)):
    P = u[:,:k[i]+1]
    result += [regreession_multi(Xtrain.dot(P),Ytrain,Xtest.dot(P),
                Ytest,0.00001)]
error = [result[i]['test_error'] for i in range(len(result))]
plt.plot(np.arange(1,len(error)+1),error)
plt.xticks(np.arange(1,len(error)+1),k)
plt.title('Mean Error in Testing vs k')
plt.show()
k[np.argmin(error)]
```

(e)

We can see that the predict image is more like a real face. Even better than the given true grayscale images.

Solution (cont.)



```
plt.figure(figsize=(30,40))
for i in range(4):
  plt.subplot(4,3,3*i+1)
  plt.imshow(x_test[i].astype("uint8"))
  plt.subplot(4,3,3*i+2)
  plt.imshow(y_test[i].astype("uint8"))
  plt.subplot(4,3,3*i+3)
  vector = ((Xtest[i,:].dot(u[:,:k[np.argmin(error)]+1]).dot(
          result[np.argmin(error)]['A'])+1.0) / 2.0) * 255.0
  vector = np.reshape(vector, (15, 15, 3),order='F')
```

Solution (cont.)

```
    p = vector.astype("uint8")
    p = resize(p, (100, 100))
    count = 0
    plt.imshow(p)
plt.show()
```

Question 5

Have uploaded to Gradescope.

Question 6

Question How to do the statistic test to CCA?

Solution

Suppose the data is $\hat{X} \in \mathbb{R}^{n\times p}, \hat{Y} \in \mathbb{R}^{n\times q}$, first pairs of canonical coefficents and varables are $(u_1, v_1) = (a^*, b^*)$ and $(U_1, V_1) = (u_1^T X, v_1^T Y)$ are given in question 3. Repeat the similar processes we can get $s = \min\{p, q\}$ pairs.

(1) Calculate the sample covariance matrix

$$\hat{\Sigma} = \begin{bmatrix} \hat{\Sigma}_{XX} & \hat{\Sigma}_{XY} \\ \hat{\Sigma}_{YX} & \hat{\Sigma}_{YY} \end{bmatrix}$$

(2) The entire test:

$$H_0 : \Sigma_{XY} = 0 \qquad\qquad H_1 : \Sigma_{XY} \neq 0$$

i.e.

$$H_0 : \lambda_1 = \lambda_2 = \cdots = \lambda_s = 0 \qquad\qquad H_1 : \lambda_i \neq 0 \quad (i = 1, 2, \cdots, s)$$

Let

$$\begin{aligned} \Lambda_1 &= \frac{|\hat{\Sigma}|}{|\hat{\Sigma}_{XX}||\hat{\Sigma}_{YY}|} \\ &= |I_p - \hat{\Sigma}_{XX}^{-1}\hat{\Sigma}_{XY}\hat{\Sigma}_{YY}^{-1}\hat{\Sigma}_{YX}| \\ &= \prod_{i=1}^{s}(1-\lambda_i^2) \end{aligned}$$

When H_0 is true,

$$Q_1 = -\left[n - 1 - \frac{1}{2}(p+q+1)\right]\ln\Lambda_1 \sim \chi^2(pq)$$

Given significance level α. If $Q_1 \geqslant \chi^2_\alpha(pq)$, then reject H_0 and conclude that at least the first pair (U_1, V_1) is linear correlated.

(3) The partial test:

$$H_0 : \lambda_k = \cdots = \lambda_s = 0 \qquad\qquad H_1 : \lambda_i \neq 0 \quad (i = k, k+1, \cdots, s)$$

When H_0 is true,

$$Q_1 = -\left[n - 1 - \frac{1}{2}(p+q+1)\right]\ln\Lambda_k \sim \chi^2((p-k+1)(q-k+1))$$

where

$$\Lambda_k = \prod_{i=k}^{s}(1-\lambda_i^2)$$

If H_0 is true, then conclude that only the first $k-1$ pairs of canonical varables are linear correlated. If not, then the kth pair of canonical varables is also linear correlated and need to proceed the $(k+1)$th test.

HW5

September 27, 2017

1 Question 4

1.1 (a)

```
In [168]: import pickle
          import numpy as np
          from scipy.linalg import eig
          from scipy.linalg import sqrtm
          from numpy.linalg import inv
          from numpy.linalg import svd
          import matplotlib.pyplot as plt
          from sklearn.preprocessing import StandardScaler
          %matplotlib inline
```

```
In [169]: with open('hw05-data/x_train.p','rb') as f:
              s = f.read()
              s = s.replace(b'\r',b'')
          with open('hw05-data/x_train.p','wb') as f:
              f.write(s)
          with open('hw05-data/y_train.p','rb') as f:
              s = f.read()
              s = s.replace(b'\r',b'')
          with open('hw05-data/y_train.p','wb') as f:
              f.write(s)
          with open('hw05-data/x_test.p','rb') as f:
              s = f.read()
              s = s.replace(b'\r',b'')
          with open('hw05-data/x_test.p','wb') as f:
              f.write(s)
          with open('hw05-data/y_test.p','rb') as f:
              s = f.read()
              s = s.replace(b'\r',b'')
          with open('hw05-data/y_test.p','wb') as f:
              f.write(s)
```

```
In [170]: with open('hw05-data/x_train.p','rb') as f:
              x_train = pickle.load(f,encoding='latin1')
          with open('hw05-data/y_train.p','rb') as f:
```

```
        y_train = pickle.load(f,encoding='latin1')
    with open('hw05-data/x_test.p','rb') as f:
        x_test = pickle.load(f,encoding='latin1')
    with open('hw05-data/y_test.p','rb') as f:
        y_test = pickle.load(f,encoding='latin1')
```

In [171]:
```
x_train = np.array(x_train)
y_train = np.array(y_train)
x_test = np.array(x_test)
y_test = np.array(y_test)
```

In [4]:
```
print(np.shape(x_train))
print(np.shape(y_train))
```

```
(956, 15, 15, 3)
(956, 15, 15, 3)
```

In [186]:
```
Xtrain = np.reshape(x_train,(len(x_train),np.prod(np.shape(x_train)[1:])),order='F')
Ytrain = np.reshape(y_train,(len(y_train),np.prod(np.shape(y_train)[1:])),order='F')
Xtrain = Xtrain/255*2.0-1.0
Ytrain = Ytrain/255*2.0-1.0
Xtest = np.reshape(x_test,(len(x_test),np.prod(np.shape(x_test)[1:])),order='F')
Ytest = np.reshape(y_test,(len(y_test),np.prod(np.shape(y_test)[1:])),order='F')
Xtest = Xtest/255*2.0-1.0
Ytest = Ytest/255*2.0-1.0
```

In [187]:
```
print(np.shape(Xtrain))
print(np.shape(Ytrain))
```

```
(956, 675)
(956, 675)
```

In [188]:
```
n, m = np.shape(Xtrain)
xmean = np.mean(Xtrain,axis=0)
ymean = np.mean(Ytrain,axis=0)
sigmaxy = (Xtrain-xmean[np.newaxis,:]).T.dot((Ytrain-ymean[np.newaxis,:]))/(n-1)
sigmaxx = (Xtrain-xmean[np.newaxis,:]).T.dot((Xtrain-xmean[np.newaxis,:]))/(n-1)
sigmayy = (Ytrain-ymean[np.newaxis,:]).T.dot((Ytrain-ymean[np.newaxis,:]))/(n-1)
```

In [189]:
```
print(np.shape(sigmaxy))
print(np.shape(sigmaxx))
print(np.shape(sigmayy))
```

```
(675, 675)
(675, 675)
(675, 675)
```

1.2 (b)

In [190]:
```
lamb = 0.00001
M = inv(sqrtm(sigmaxx+lamb)).dot(sigmaxy).dot(inv(sqrtm(sigmayy+lamb)))
[u, s, vt] = svd(M)
```

In [191]:
```
plt.figure(figsize=(40,20))
plt.plot(s)
plt.xticks(np.arange(1,m+1,10))
plt.title('Spectrum of singular values')
plt.show()
```

In [192]:
```
print(s)
```

```
[  2.55050688e+18   1.25086110e+18   6.94562406e+17   7.46011196e+05
   4.64767241e+05   3.45506326e+05   2.71355857e+05   2.01784825e+05
   1.86396124e+05   1.71170650e+05   1.66204183e+05   1.59149509e+05
   1.56393825e+05   1.53735895e+05   1.47337443e+05   1.46908503e+05
   1.41447123e+05   1.40952773e+05   1.38287438e+05   1.33313727e+05
   1.32784352e+05   1.31491894e+05   1.30069807e+05   1.28753587e+05
   1.26265879e+05   1.22894346e+05   1.22415628e+05   1.22349022e+05
   1.19604201e+05   1.15425844e+05   1.15211602e+05   1.12938294e+05
   1.12050258e+05   1.11859666e+05   1.10280574e+05   1.09903935e+05
   1.08664866e+05   1.04267577e+05   1.04258723e+05   1.02403969e+05
   1.02007360e+05   1.00437442e+05   9.91436303e+04   9.90820915e+04
   9.78129507e+04   9.63560816e+04   9.54010934e+04   9.42372105e+04
   9.35433453e+04   9.33161612e+04   9.16242753e+04   9.15519525e+04
   8.99578672e+04   8.98462513e+04   8.88771166e+04   8.73739518e+04
   8.68563792e+04   8.59526958e+04   8.53271006e+04   8.47621909e+04
```

8.38286683e+04	8.28012390e+04	8.15724090e+04	8.14805127e+04
8.03216568e+04	8.00980814e+04	7.98897686e+04	7.88831116e+04
7.88456611e+04	7.79279328e+04	7.76954153e+04	7.63668126e+04
7.59136021e+04	7.57321602e+04	7.49312396e+04	7.48472102e+04
7.42575870e+04	7.33902653e+04	7.29936439e+04	7.27088607e+04
7.18516120e+04	7.11722373e+04	7.04151617e+04	7.03945093e+04
6.99042564e+04	6.98699163e+04	6.94957701e+04	6.88037408e+04
6.79005801e+04	6.78946719e+04	6.75832071e+04	6.62873810e+04
6.57605885e+04	6.52351630e+04	6.51172067e+04	6.46989425e+04
6.44560980e+04	6.37290957e+04	6.35038775e+04	6.25658526e+04
6.16321576e+04	6.10489917e+04	6.07555051e+04	6.05294410e+04
6.02471659e+04	5.97070541e+04	5.95304154e+04	5.85883310e+04
5.80321232e+04	5.74051953e+04	5.72581472e+04	5.63961152e+04
5.61881915e+04	5.58169784e+04	5.55158549e+04	5.49569131e+04
5.49323525e+04	5.44722199e+04	5.43160791e+04	5.40418559e+04
5.34585627e+04	5.32454288e+04	5.28247347e+04	5.22989876e+04
5.20287568e+04	5.14959023e+04	5.12548376e+04	5.10107745e+04
5.08165004e+04	5.03370584e+04	5.01373626e+04	4.98933958e+04
4.91734006e+04	4.90188789e+04	4.83154215e+04	4.80828113e+04
4.77740888e+04	4.71830001e+04	4.68705015e+04	4.67454308e+04
4.65314643e+04	4.58634597e+04	4.57673807e+04	4.56416305e+04
4.51464326e+04	4.49440808e+04	4.49273313e+04	4.48368197e+04
4.45292260e+04	4.42466681e+04	4.40110397e+04	4.36895615e+04
4.35549611e+04	4.30103980e+04	4.22084485e+04	4.21684687e+04
4.20277686e+04	4.20183708e+04	4.15048969e+04	4.14243903e+04
4.13549525e+04	4.10352225e+04	4.09587621e+04	4.04718910e+04
4.02081573e+04	4.01864158e+04	3.95804138e+04	3.95305042e+04
3.88841913e+04	3.86311925e+04	3.84616542e+04	3.82432177e+04
3.81490598e+04	3.76783903e+04	3.72887814e+04	3.71206661e+04
3.67819551e+04	3.67029147e+04	3.63451973e+04	3.61405482e+04
3.61097160e+04	3.60810022e+04	3.56415499e+04	3.54246257e+04
3.50480045e+04	3.49235736e+04	3.49125142e+04	3.46658852e+04
3.44413944e+04	3.39048144e+04	3.37808594e+04	3.37150261e+04
3.31308184e+04	3.30813300e+04	3.28721137e+04	3.25328693e+04
3.22296134e+04	3.22267176e+04	3.21254048e+04	3.19712944e+04
3.15777023e+04	3.15238663e+04	3.13470855e+04	3.11375114e+04
3.09462233e+04	3.06772557e+04	3.03902971e+04	3.03037302e+04
3.02836941e+04	3.02355371e+04	2.99038130e+04	2.98738159e+04
2.94485081e+04	2.92788017e+04	2.90161201e+04	2.88889265e+04
2.86888278e+04	2.85072334e+04	2.82892914e+04	2.81460245e+04
2.78752981e+04	2.77042336e+04	2.73947434e+04	2.73511519e+04
2.72770346e+04	2.70063921e+04	2.68956999e+04	2.66029474e+04
2.64259505e+04	2.63438214e+04	2.61545684e+04	2.60217554e+04
2.58448590e+04	2.56190775e+04	2.55673781e+04	2.53558758e+04
2.53433250e+04	2.50729331e+04	2.50039042e+04	2.46062216e+04
2.45903984e+04	2.44364848e+04	2.41613482e+04	2.39207755e+04
2.38279734e+04	2.36999806e+04	2.36566444e+04	2.33730952e+04
2.32539158e+04	2.30791593e+04	2.28953003e+04	2.28250823e+04

2.25829707e+04	2.25315028e+04	2.23415857e+04	2.21967260e+04
2.19846156e+04	2.18837786e+04	2.17194182e+04	2.15774402e+04
2.14456857e+04	2.13403182e+04	2.11256491e+04	2.10582412e+04
2.09183724e+04	2.08954777e+04	2.08593793e+04	2.06804916e+04
2.04620858e+04	2.02077594e+04	2.00655609e+04	1.99922184e+04
1.99562028e+04	1.96111097e+04	1.95775716e+04	1.94200650e+04
1.92430409e+04	1.91062589e+04	1.90701156e+04	1.89145980e+04
1.87427990e+04	1.85117677e+04	1.84324522e+04	1.83934905e+04
1.82942893e+04	1.81674226e+04	1.78263663e+04	1.76222988e+04
1.75068707e+04	1.73572717e+04	1.72403941e+04	1.72222710e+04
1.70866657e+04	1.69727875e+04	1.68222850e+04	1.67464248e+04
1.65577706e+04	1.64944742e+04	1.63799164e+04	1.62871250e+04
1.61965699e+04	1.61315495e+04	1.58787898e+04	1.57185178e+04
1.55873627e+04	1.55751996e+04	1.55050904e+04	1.54748105e+04
1.51995317e+04	1.51969409e+04	1.51430240e+04	1.49900596e+04
1.48779338e+04	1.48147868e+04	1.46200338e+04	1.44598944e+04
1.44260411e+04	1.43008250e+04	1.42163200e+04	1.40839174e+04
1.39536310e+04	1.37779229e+04	1.36011297e+04	1.34382287e+04
1.34057041e+04	1.32422554e+04	1.31698543e+04	1.30793826e+04
1.29906424e+04	1.28777617e+04	1.27714442e+04	1.27141529e+04
1.26499085e+04	1.24124459e+04	1.23814179e+04	1.22494858e+04
1.22030456e+04	1.20928066e+04	1.19561949e+04	1.17639191e+04
1.16870235e+04	1.16812821e+04	1.15568322e+04	1.14905400e+04
1.14139612e+04	1.13167699e+04	1.12497465e+04	1.10826992e+04
1.09877891e+04	1.08704675e+04	1.07770939e+04	1.07747740e+04
1.06864390e+04	1.05888067e+04	1.04925174e+04	1.03729781e+04
1.02572174e+04	1.01089960e+04	1.00091433e+04	9.96341205e+03
9.80406067e+03	9.65619747e+03	9.62609861e+03	9.58124638e+03
9.40369622e+03	9.34590803e+03	9.20404907e+03	8.97483830e+03
8.92484694e+03	8.83876291e+03	8.82047972e+03	8.79059004e+03
8.67701692e+03	8.64140166e+03	8.52711739e+03	8.41215521e+03
8.25567101e+03	8.15204580e+03	8.01814312e+03	7.93226354e+03
7.89378656e+03	7.82751505e+03	7.72773821e+03	7.62687215e+03
7.56075738e+03	7.45612051e+03	7.38997418e+03	7.21767740e+03
7.10967315e+03	7.02952172e+03	6.99921130e+03	6.96593094e+03
6.83202955e+03	6.72903679e+03	6.48936182e+03	6.36431932e+03
6.35245262e+03	6.32577422e+03	6.22393459e+03	6.13679936e+03
5.97075354e+03	5.88515934e+03	5.83170275e+03	5.69089150e+03
5.57429471e+03	5.45700205e+03	5.34152806e+03	5.14743591e+03
5.13917870e+03	4.81235291e+03	4.62272962e+03	4.31281784e+03
4.04231751e+03	3.35684067e+03	3.18848159e+03	2.29772223e+03
1.64184254e+03	6.73266436e+02	6.23213377e+02	5.17747020e+02
5.10007444e+02	4.96246099e+02	4.92800628e+02	4.32568603e+02
4.00006353e+02	3.80721382e+02	3.53798412e+02	3.17620782e+02
2.67399007e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02

2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02
2.45567303e+02	2.45567303e+02	2.45567303e+02	2.45567303e+02

```
2.45567303e+02   2.45567303e+02   2.45567303e+02   2.45567303e+02
2.45567303e+02   2.45567303e+02   2.45567303e+02   2.45567303e+02
2.45567303e+02   2.45567303e+02   2.45567303e+02   2.45567303e+02
2.45567303e+02   2.45567303e+02   2.45567303e+02   2.45567303e+02
2.45567303e+02   2.45567303e+02   2.45567303e+02   2.45567303e+02
2.45567303e+02   2.45567303e+02   2.45567303e+02   2.45567303e+02
2.45567303e+02   2.45567303e+02   2.45567303e+02   2.45567303e+02
2.45567303e+02   2.45567303e+02   2.45567303e+02   2.45567303e+02
2.45567303e+02   2.45567303e+02   2.28703528e+02   1.81387758e+02
1.29928617e+02   8.05804383e+01   2.80926753e+01]
```

1.3 (c)

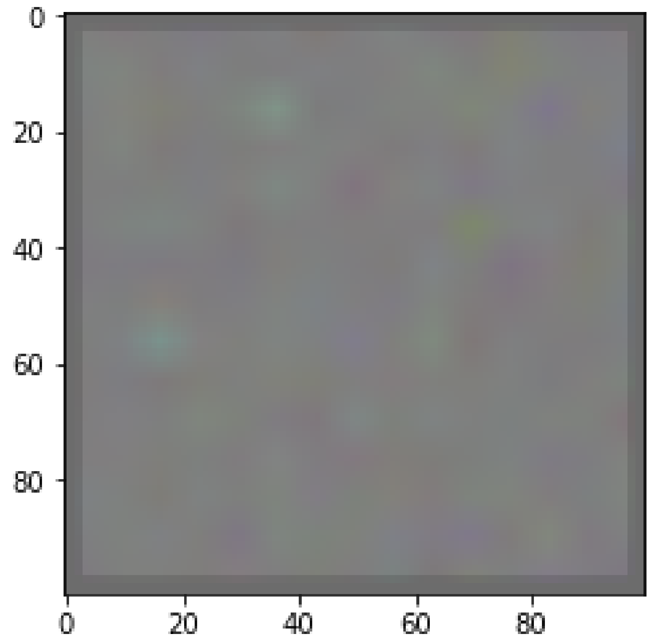
```
In [118]: np.shape(u)
```

```
Out[118]: (675, 675)
```

```
In [194]: from skimage.transform import resize
          from skimage.io import imsave

          def plot_image(vector):
              vector = ((vector+1.0) / 2.0) * 255.0
              vector = np.reshape(vector, (15, 15, 3),order='F')
              p = vector.astype("uint8")
              p = resize(p, (100, 100))
              plt.imshow(p)
              imsave('eigenface.png',p)
```

```
In [195]: plot_image(u[:,0])
```

```
/anaconda/lib/python3.6/site-packages/ipykernel_launcher.py:7: ComplexWarning: Casting complex v
  import sys
/anaconda/lib/python3.6/site-packages/skimage/transform/_warps.py:84: UserWarning: The default m
  warn("The default mode, 'constant', will be changed to 'reflect' in "
/anaconda/lib/python3.6/site-packages/skimage/io/_io.py:132: UserWarning: eigenface.png is a low
  warn('%s is a low contrast image' % fname)
/anaconda/lib/python3.6/site-packages/skimage/util/dtype.py:122: UserWarning: Possible precision
  .format(dtypeobj_in, dtypeobj_out))
```



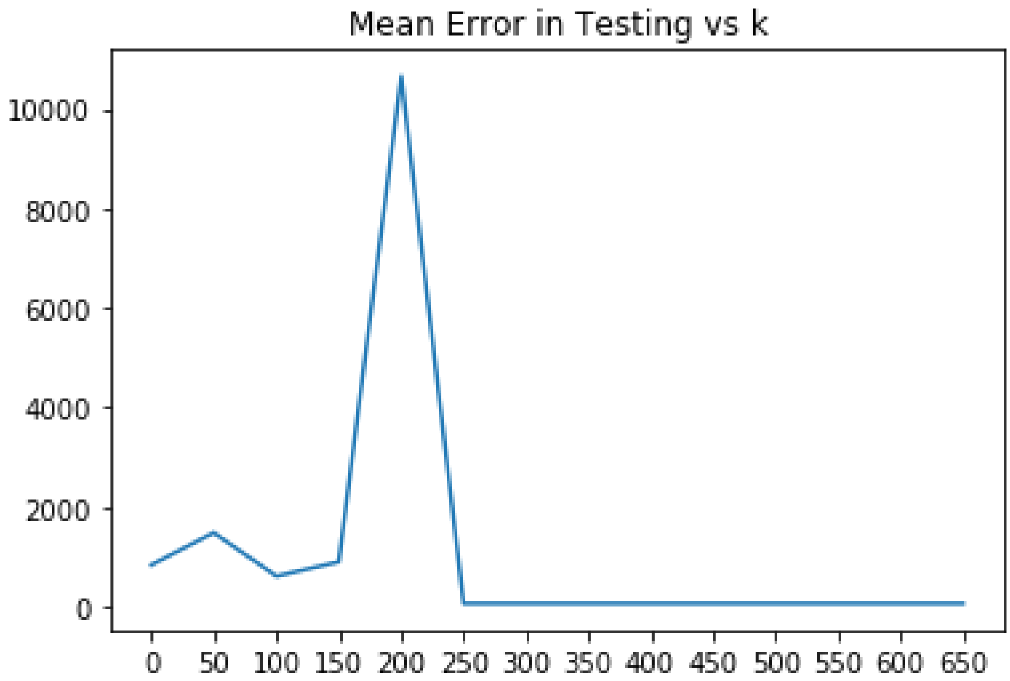
1.4 (d)

```
In [149]: def regreession_multi(X,y,xtest,ytest,lamb):
              if len(np.shape(X))<2:
                  X= X[:,np.newaxis]
              if len(np.shape(xtest))<2:
                  xtest= xtest[:,np.newaxis]
              n1, n2 = np.shape(X)
              A = np.linalg.solve(X.T.dot(X)+lamb*np.identity(n2),X.T.dot(y))
              yhat = X.dot(A)

              Rmean =  np.linalg.norm(yhat-y)**2/n1

              yhat_test = xtest.dot(A)
              Rmean_test = np.linalg.norm(yhat_test-ytest)**2/len(ytest)

              return {'A':A,'train_error':Rmean,'test_error':Rmean_test}
```

```
In [206]: k = np.arange(0,700,50)
          result= []
          for i in range(len(k)):
              P = u[:,:k[i]+1]
              result += [regreession_multi(Xtrain.dot(P),Ytrain,Xtest.dot(P),Ytest,0.00001)]
          error = [result[i]['test_error'] for i in range(len(result))]
```

```
In [211]: plt.plot(np.arange(1,len(error)+1),error)
          plt.xticks(np.arange(1,len(error)+1),k)
```

```
plt.title('Mean Error in Testing vs k')
plt.show()
k[np.argmin(error)]
```



Out[211]: 450

In [208]: error

```
Out[208]: [836.10560817291821,
 1484.4194009178789,
 611.01410485769702,
 897.03636376498798,
 10663.708924755083,
 61.013945886510314,
 61.013933403551626,
 61.013937941094973,
 61.013944115214649,
 61.013929325229412,
 61.013939874027159,
 61.013939659671792,
 61.013937415266533,
 61.013939069869295]
```

1.5 (e)

```
In [217]: plt.figure(figsize=(30,40))
          for i in range(4):
```

```
            plt.subplot(4,3,3*i+1)
            plt.imshow(x_test[i].astype("uint8"))
            plt.subplot(4,3,3*i+2)
            plt.imshow(y_test[i].astype("uint8"))
            plt.subplot(4,3,3*i+3)
            vector = ((Xtest[i,:].dot(u[:,:k[np.argmin(error)]+1]).dot(result[np.argmin(error)
            vector = np.reshape(vector, (15, 15, 3),order='F')
            p = vector.astype("uint8")
            p = resize(p, (100, 100))
            count = 0
            plt.imshow(p)
        plt.show()
```

```
/anaconda/lib/python3.6/site-packages/ipykernel_launcher.py:10: ComplexWarning: Casting complex
  # Remove the CWD from sys.path while we load stuff.
/anaconda/lib/python3.6/site-packages/skimage/transform/_warps.py:84: UserWarning: The default m
  warn("The default mode, 'constant', will be changed to 'reflect' in "
```