10725/36725 Optimization
Homework 3

Due October 23, 2012 at beginning of class

Instructions: There are four questions in this assignment. Please submit your homework as 4 separate sets of pages with your name and userid on each set. For the last question which involves coding, please print out your code and graphs and attach them to the written part of your homework and email your code to submission10725f12@gmail.com. Refer to the course webpage for policies regarding collaboration, due dates, and extensions.

1 From Vector To Matrix Calculus [Thanks to Abhinav Shrivastava]

1.1 PCA [10 points]

In this subsection we will warm up our vector calculus skills by showing two cool ways of interpreting the first principal component. One is the direction of maximum variance after projection and the second is the direction that minimizes reconstruction error. Note that the first principal component is the first eigenvector of the sample covariance matrix. Consider n points $X_1, ..., X_n$ in p-dimensional space, and let X be the $n \times p$ matrix representing these points. Assume that the data points are centered, ie, $\vec{1}^\top X = \vec{0}$. Consider a unit vector $v \in \mathbb{R}^p$ and project all the points onto this vector (hence every point becomes a one-dimensional point on the direction of unit vector v).

1. Argue that the projection is given by Xv.

[Solution] Let us decompose the vector representing X_i into two orthogonal vectors X_{iv} and $X_{iv'}$, where X_{iv} is parallel to v. Using notation $X = [X_1^T; X_2^T; \cdots X_n^T] = [X_i^T]$, we can write

$$Xv = \left[(X_{iv} + X_{iv'})^T\right] v = [X_{iv}^T]v + [X_{iv'}^T]v = [X_{iv}^T]v = X_v v$$

Given that v is a unit vector, $X_v v$ gives the component of X in direction v. Since $Xv = X_v v$,

Xv represents projection of X onto v. ★

---

2. What is the sample mean of all the points after the projection?

[Solution] Sample mean after projection is given by

$$\frac{1}{n}[\vec{1}^\top(Xv)] = \frac{1}{n}[\vec{1}^\top(X)v] = \frac{1}{n}[\vec{0}^\top v] = 0$$

Hence, the sample mean is 0. ★

---

3 What is the sample variance of all the points after the projection?

[Solution] Given sample mean is 0, sample variance after projection is given by

$$\frac{1}{n}[(Xv)^T(Xv)] = \frac{1}{n}[v^T X^T X v] = v^T \left[\frac{X^T X}{n}\right] v = v^T \Sigma v$$

where $\Sigma = X^T X$ is the sample variance of original p-dimensional points (X). ★

---

4 Setup the problem of maximizing the sample variance of the projection onto v subject to a constraint on the L2-norm of v.

[Solution] The problem of finding unit vector v that maximizes sample variance can be written as

$$\max_{v} v^T \Sigma v$$
$$\text{s.t. } \|v\|^2 = 1$$

The lagrangian of this problem can be written as

$$\mathcal{L}(v, \lambda) = v^T \Sigma v - \lambda(v^T v - 1)$$

★

---

5 Solve using vector derivatives to show that the solution is the first PC.

[Solution] By stationarity, at optimality we have

$$2(\Sigma v^*) - 2\lambda^* v^* = 0$$

The optimal value is $v^{*T} S v^* = \lambda^* v^{*T} v^* = \lambda^*$, and so the vector that maximizes variance after projection, is the **eigenvector** associated with the largest **eigenvalue** λ of the covariance matrix Σ, which by definition is the first Principal Component (PC). ★

So we have now proved that the direction of maximum covariance is the first PC. Now we show that the direction that minimizes reconstruction error is also the first PC.

1. Argue that the reconstruction of X_i using v is $(X_i^\top v)v$.

[Solution] The reconstruction of X_i using v can be written as the following optimization problem (with α being a scalar): $\min_\alpha \|X_i - \alpha v\|^2$. Taking derivative w.r.t α and setting it to 0 gives us the following:

$$2(X_i - \alpha v)^T v = 0 \Leftrightarrow X_i^T v = \alpha v^T v \Leftrightarrow \alpha = X_i^T v$$

Since v is a unit vector, $v^T v = 1$. So, $\alpha v = (X_i^T v)v$ is the reconstruction of X_i using v. ★

---

2. You projected X_i to $X_i^\top v$ and then reconstructed it using $(X_i^\top v)v$. What is the reconstruction error of X_i, when measured in L2-norm?

[Solution] Reconstruction error in $L2$-norm is $\|(X_i^T v)v - X_i\|_2$ ★

3. What is the total squared reconstruction error over all points?

[Solution] Total squared reconstruction error over all points: $\|(Xv)v^T - X\|_F^2$ ★

4. Show that minimizing total squared reconstruction error is equivalent to minimizing $-\|Xv\|_2^2$.

[Solution] Total squared reconstruction error can be written as (let $\hat{X} = (Xv)v^T$)

$$\begin{aligned}
\|\hat{X} - X\|_F^2 &= \text{tr}\left((\hat{X} - X)^T(\hat{X} - X)\right) \\
&= \text{tr}\left(\hat{X}^T\hat{X} - 2\hat{X}^T X + X^T X\right) \\
&= \text{tr}\left(\hat{X}^T\hat{X}\right) - \text{tr}\left(2\hat{X}^T X\right) + \text{tr}\left(X^T X\right) \\
&= \text{tr}\left((Xvv^T)^T(Xvv^T)\right) - \text{tr}\left(2(Xvv^T)^T X\right) + \text{tr}\left(X^T X\right) \\
&= \text{tr}\left(vv^T X^T Xvv^T\right) - \text{tr}\left(2vv^T X^T X\right) + \text{tr}\left(X^T X\right) \\
&= \text{tr}\left(v^T X^T Xvv^T v\right) - 2\text{tr}\left(v^T X^T Xv\right) + \text{tr}\left(X^T X\right) \\
&= \text{tr}\left(v^T X^T Xv\right) - 2\text{tr}\left(v^T X^T Xv\right) + \text{tr}\left(X^T X\right) \\
&= -\text{tr}\left((Xv)^T(Xv)\right) + \text{tr}\left(X^T X\right) \\
&= -\|Xv\|_2^2 + \|X\|_2^2
\end{aligned}$$

Since, $\|X\|^2$ is a constant w.r.t. v, minimizing total squared reconstruction error $\|(Xv)v^T - X\|_F^2$ is equivalent to minimizing $-\|Xv\|_2^2$. ★

5. Solve using vector derivatives to show that the solution is the first PC.

[Solution] Minimizing $-\|Xv\|_2^2$ is equivalent to maximizing $v^T X^T Xv$, same as part 5 of previous subsection. ★

1.2 Matrix Factorization [15 points]

As its name suggests, matrix factorization involves factoring a matrix into a product of two other matrices. The SVD, LU and QR decomposition are examples of important types of factorizations. In some settings, we might postulate that a given $m \times n$ matrix Y of noisy values is actually low rank . One way to write this as an optimization problem (where $\|.\|_F$ is the frobenius norm, $\|.\|_*$ is the nuclear norm) is as follows:

$$L(Z) = \min_Z \|Y - Z\|_F^2 + \lambda\|Z\|_*$$

SVD Prove, using the subdifferential of the nuclear norm, that the solution to the above equation is achieved via SVD by subtracting λ from all singular values, and then setting negative ones to zero.

[Solution] Let is consider this equation (referred to the TA reg. possible inclusion of $\frac{1}{2}$):

$$L(Z) = \min_Z \frac{1}{2}\|Z - Y\|_F^2 + \lambda\|Z\|_* \tag{1}$$

Let SVD$(Y) = U\Sigma V^T$. We can decompose U, V, Σ as following:

$$\text{SVD}(Y) = U\Sigma V^T = U_0\Sigma_0 V_0^T + U_1\Sigma_1 V_1^T$$

where, where U_0, V_0 (and respectively U_1, V_1) are the singular vectors associated with singular values smaller than or equal to λ (and respectively greater than λ) (singular value thresholding). To minimize (1) w.r.t Z, we should have $\partial L/\partial Z = 0$, i.e.

$$\begin{aligned}(Z - Y) + \lambda\ \partial\|Z\|_* &= 0\\ (Y - Z) &= \lambda\ \partial\|Z\|_*\end{aligned} \tag{2}$$

Let us select Z^* by taking SVD ($\underset{2}{=}$ in (3)) of Y and subtracting λ from all singular values ($\underset{2}{=}$ in (3)), and then setting negative ones to zero ($\underset{3}{\Rightarrow}$ in (3)), as all singular values in Σ_0 are $\leq \lambda$, $\therefore (\Sigma_0 - \lambda) \leq 0$, hance setting those to 0.

$$Z^* \underset{1}{=} U(\Sigma - \lambda)V^T \underset{2}{=} U_0(\Sigma_0 - \lambda)V_0^T + U_1(\Sigma_1 - \lambda)V_1^T \underset{3}{\Rightarrow} U_1(\Sigma_1 - \lambda)V_1^T \tag{3}$$

Let us prove that Z^* shown above is a valid solution for equation (2), hence a solution for (1). Plugging in values of Y and Z^*, we can rewrite L.H.S. of (2) as

$$\begin{aligned}(Y - Z^*) &= U_0\Sigma_0 V_0^T + U_1\Sigma_1 V_1^T - U_1(\Sigma_1 - \lambda)V_1^T\\ &= U_0\Sigma_0 V_0^T + U_1\Sigma_1 V_1^T - U_1\Sigma_1 V_1^T + \lambda U_1 V_1^T\\ &= U_0\Sigma_0 V_0^T + \lambda U_1 V_1^T\\ &= \lambda\left(U_1 V_1^T + \frac{1}{\lambda}U_0\Sigma_0 V_0^T\right)\end{aligned}$$

$$(Y - Z^*) = \lambda\left(U_1 V_1^T + W_1\right) \tag{4}$$

where $W_1 = \left(\frac{1}{\lambda}U_0\Sigma_0 V_0^T\right)$. Recall from HW2, the sub-differential of nuclear norm is given by:

$$\partial||A||_* = \{UV^T + W : U^T W = 0, WV = 0, ||W||_2 \leq 1, \} \tag{5}$$

The L.H.S. of (2) is given in (4). Let us show that this (4) is equal to $\lambda\times$ sub-differential of nuclear norm of Z (given by RHS of (2)). Given the forms in (5) and (4), we need to show that $U^T W = 0, WV = 0, ||W||_2 \leq 1$.

1. $U_1^T W_1 = U_1^T \left(\frac{1}{\lambda} U_0 \Sigma_0 V_0^T\right) = \left(\frac{1}{\lambda}(U_1^T U_0)\Sigma_0 V_0^T\right) = 0$ (orthonormal basis of SVD).
2. $W_1 V_1 = \left(\frac{1}{\lambda} U_0 \Sigma_0 V_0^T\right) V_1 = \left(\frac{1}{\lambda} U_0 \Sigma_0 (V_0^T V_1)\right) = 0$ (orthonormal basis of SVD).
3. $\|W_1\|_2 = \frac{1}{\lambda}\|U_0 \Sigma_0 V_0^T\| \leq \frac{1}{\lambda}\|U_0\|\|\Sigma_0\|\|V_0^T\| \leq \frac{1}{\lambda}\|\Sigma_0\| \leq 1$ (all values in $\Sigma_0 \leq \lambda$, $\therefore$ $\|\Sigma_0\| \leq \lambda$).

Therefore, U_1, V_1 and W_1 given in (4) follow the properties of general $\partial\|A\|_*$ (as in (5)). Therefore, we can write $U_1 V_1^T + W_1 = \partial\|Z\|_*$. Now, (4) becomes:

$$(Y - Z^*) = \lambda \ \left(U_1 V_1^T + W_1\right) = \lambda \ \partial\|Z\|_* = \text{R.H.S of (2)} \tag{6}$$

Therefore, the Z^* selected in (3) actually solves (2) and hence minimizes (1). ★

Since Z is postulated to be low rank, it can be decomposed into $Z = UV^\top$ where $U \in \mathbb{R}^{m\times k}$ and $V \in \mathbb{R}^{n\times k}$ for $k \geq rank(Z)$. Interestingly, here is a cool variational characterization of the nuclear norm (for a $k \geq rank(Z)$)

$$\|Z\|_* = \frac{1}{2} \min_{\{U\in\mathbb{R}^{m\times k}, V\in\mathbb{R}^{n\times k}: Z=UV^\top\}} \|U\|_F^2 + \|V\|_F^2$$

This allows us to re-pose the problem as:

$$\min_{\{U\in\mathbb{R}^{m\times k}, V\in\mathbb{R}^{n\times k}\}} L(U,V)$$

where

$$L(U,V) = \|Y - UV^\top\|_F^2 + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2)$$

1. Noting that $\|A\|_F^2 = \sum_{i,j} A_{ij}^2$, find $\partial L/\partial U_{ij}$

[Solution] Notation: U_{ij} is a scalar, U_i is a row-vector (i^{th} row), $U_{\bullet i}$ is a column-vector ((i^{th} column)).

$$\begin{aligned} L(U,V) &= \|Y - UV^\top\|_F^2 + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2) \\ &= \sum_{i,a}\left(Y_{ia} - \sum_j U_{ij}V_{aj}\right)^2 + \frac{\lambda}{2}\left(\sum_{ij} U_{ij}^2 + \sum_{ij} V_{ij}^2\right) \end{aligned}$$

$$\begin{aligned}
\frac{\partial L(U,V)}{\partial U_{ij}} &= \frac{\partial}{\partial U_{ij}}\left[\sum_{i,a}\left(Y_{ia}-\sum_j U_{ij}V_{aj}\right)^2+\frac{\lambda}{2}\left(\sum_{ij}U_{ij}^2+\sum_{ij}V_{ij}^2\right)\right] \\
&= 2\sum_a\left(\left(Y_{ia}-\sum_j U_{ij}V_{aj}\right)(-V_{aj})\right) \;+\; \frac{\lambda}{2}(2U_{ij}) \\
&= 2\sum_a\left(\left(Y_{ia}-U_iV_a^T\right)(-V_{aj})\right) \;+\; \lambda\, U_{ij} \\
&= -2\left(Y_i-U_iV^T\right)(V_{\bullet j}) \;+\; \lambda\, U_{ij}
\end{aligned}$$

★

---

2. Noting that $\|A\|_F^2=\sum_i A_i^\top A_i$, find $\partial L/\partial U_i$

[Solution]

$$\begin{aligned}
L(U,V) &= \|Y-UV^\top\|_F^2+\frac{\lambda}{2}(\|U\|_F^2+\|V\|_F^2) \\
&= \sum_i\left(\left(Y_i-U_iV^T\right)^T\left(Y_i-U_iV^T\right)\right)+\frac{\lambda}{2}\left(\sum_i U_i^TU_i+\sum_i V_i^TV_i\right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial L(U,V)}{\partial U_i} &= \frac{\partial}{\partial U_i}\left[\sum_i\left(\left(Y_i-U_iV^T\right)^T\left(Y_i-U_iV^T\right)\right)+\frac{\lambda}{2}\left(\sum_i U_i^TU_i+\sum_i V_i^TV_i\right)\right] \\
&= 2\sum_i\left(Y_i-U_iV^T\right)\frac{\partial(Y_i-U_iV^T)^T}{\partial U_i}+\lambda\, U_i \\
&= -2\left(Y_i-U_iV^T\right)V+\lambda\, U_i
\end{aligned}$$

★

---

3. Noting that $\|A\|_F^2 = Tr(A^\top A)$, find $\partial L/\partial U$ and $\partial L/\partial V$

[Solution]

$$
\begin{aligned}
L(U,V) &= \|Y - UV^\top\|_F^2 + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2) \\
&= \mathrm{Tr}\left(\left(Y - UV^T\right)^T \left(Y - UV^T\right)\right) + \frac{\lambda}{2}\left(\mathrm{Tr}(U^TU) + \mathrm{Tr}(V^TV)\right) \\
\frac{\partial L(U,V)}{\partial U} &= \frac{\partial}{\partial U_i}\left[\mathrm{Tr}\left(\left(Y - UV^T\right)^T \left(Y - UV^T\right)\right) + \frac{\lambda}{2}\left(\mathrm{Tr}(U^TU) + \mathrm{Tr}(V^TV)\right)\right] \\
&= 2\left(Y - UV^T\right)\left[\frac{\partial}{\partial U}\left(Y - UV^T\right)\right]^T + \lambda\, U \\
&= 2\left(Y - UV^T\right)V + \lambda\, U
\end{aligned}
$$

Similarily,

$$
\frac{\partial L(U,V)}{\partial V} = 2\left(Y - UV^T\right)^T U + \lambda\, V
$$

★

Note that L is convex in $Z = UV^\top$, but non-convex in U, V. However, we could conceive of an alternate EM-like way of getting a reasonable solution to $L(U,V)$. We could alternate between fixing V and doing a gradient descent on U, and then fixing U and doing a gradient descent on V. So, keep in mind that different ways of posing an optimization problem can open different avenues to solving it.

2 Dual with Dual Duo

2.1 Round 1

(a) Is (2) a convex program? Why or why not?

This is not a convex optimization problem due to constraining a_v and b_v^f to belong to the non-convex set $\mathcal{I}$.

(b) Take $A(a,b)$ and form the Lagrangian $L(\delta,a,b)$ by introducing Lagrange multipliers $\{\delta_{v,s}^f\}_{f\in F, v\in f, s\in S}$ for the first constraint line in (2).

$$
\begin{aligned}
L(\delta,a,b) = \sum_{v\in V}\sum_{s\in S}\phi_v(s)a_{v,s} + \sum_{f\in F}\sum_{\{s_v\}_{v\in f}\subset S}\psi_f(\{s_v\}_{v\in f})\prod_{v\in f} b_{v,s_v}^f \\
+ \sum_{f\in F}\sum_{v\in f}\sum_{s\in S}\delta_{v,s}^f(b_{v,s}^f - a_{v,s})
\end{aligned}
\tag{7}
$$

(c) What is the relationship between (2) and (3)?

Optimization problems (2) and (3) are equivalent in that they have the same optimal value. To see this, note that when maximizing over a and b, we will always choose $b^f_{v,s} = a_{v,s}$, if possible, because if we do not then we will obtain $-\infty$ by minimizing over δ.

(d) What is the relationship between (3) and (4)?

Optimization problem (4) is the dual of (3) and therefore an upper bound; we have $\max_{a,b} \min_\delta L(\delta, a, b) \leq \min_\delta \max_{a,b} L(\delta, a, b)$. To see this, let a^* and b^* be solutions to (3) and let δ^* be the solution to (4); we have

$$\max_{a,b} \min_\delta L(\delta, a, b) = \min_\delta L(\delta, a^*, b^*) \leq L(\delta^*, a^*, b^*) \leq \max_{a,b} L(\delta^*, a, b) = \min_\delta \max_{a,b} L(\delta, a, b)$$

(e) Is L convex in δ? Why or why not?

L is convex in δ since the dual problem is always convex (or concave); see class notes on duality. In particular, it is the pointwise maximum of linear functions in δ.

(f) Give a procedure (not a big formula or matrix, please) which calculates a subgradient

$$g = \{g^f_{v,s}\}_{f \in F, v \in f, s \in S}$$

of L at δ. Describe, in English, what the subgradient qualitatively represents; which $\bar{x}$ and $\{\bar{y}^f\}_{f \in F}$ make the subgradient's norm small or large?

We can construct a subgradient of the function $L(\delta)$ using the subgradient rule for pointwise maximum; $\partial(\max_i f_i(x)) = \partial f_k(x)$ where $f_k(x)$ is such that $f_k(x) = f(x)$. In particular, first we maximize Φ_v and Ψ_f to get $\bar{x}$ and $\{\bar{y}^f\}_{f \in F}$. Then, we add 1 to $g^f_{v,s}$ if $\bar{x}_v = s$ and we subtract 1 from $g^f_{v,s}$ if $\bar{y}^f_v = s$.

Conceptually, this subgradient represents the agreement between x and y on the assignment of a value s to a variable v. If $\bar{x}$ and $\{\bar{y}^f\}_{f \in F}$ have similar assignments then the subgradient's norm will be small; if they have very different assignments, theubgadient's norm will be large.

(g) Suppose we had skipped the indicator variable transformation (2) and instead formed the dual from the Lagrangian of $A(x, y)$. What would the program corresponding to (5) be? Speculate on why (5) might be 'better.'

With this approach we would have had

$$\min_\delta \; L(\delta) = \sum_{v \in V} \max_{x_v \in S} \Phi_v(x_v) + \sum_{f \in F} \max_{\{y^f_v\}_{v \in f} \subset S} \Psi_f(\{y^f_v\}_{v \in f}) \tag{8}$$

where

$$\Phi_v(x_v) = \phi_v(x_v) + x_v \sum_{f: v \in f} \delta_v^f$$
$$\Psi_f(\{y_v^f\}_{v \in f}) = \psi_f(\{y_v^f\}_{v \in f}) - \sum_{v \in f} y_v^f \delta_v^f$$

In this approach, the scaling of the subgradient depends on the difference between x_v and y_v^f. This may not be desirable since we defined S to be a structureless set. In fact, mathematical operations on S (multiplication, addition) may not be defined over S which would make the Lagrangian meaningless. We avoid these problems by using indicator variables.

2.2 Round 2

(h) Write this as a linear program. How many constraints are there?

Note that S is a finite set and that μ_v and ν^f are discrete distributions; we can write their expectations as

$$\begin{aligned} \mathop{\mathbb{E}}_{x_v \sim \mu_v}[\phi(x_v)] &= \sum_{s \in S} \mu_v(s)\phi(s) \\ \mathop{\mathbb{E}}_{y^f \sim \nu^f}[\psi(\{y_v^f\}_{v \in f})] &= \sum_{\{s_v\}_{v \in f} \in S^{|f|}} \nu^f(\{s_v\}_{v \in f})\psi(\{s_v\}_{v \in f}) \end{aligned} \tag{9}$$

where $|f|$ denotes the cardinality of f, $\mu_v(s)$ denotes the probability of s under the distribution μ_v and $\nu^f(\{s_v\}_{v \in f})$ denotes the probability of $\{s_v\}_{v \in f}$ under the distribution ν^f. For the the sake of brevity, we write $\{s_v\}$ for $\{s_v\}_{v \in f}$ when the set v is indexes is clear from context.

Using the explicit form of the expectation, we can write the optimization problem as a linear program

$$\begin{aligned} \underset{\substack{\{\mu_v\}_{v \in V} \\ \{\nu^f\}_{f \in F}}}{\text{maximize}} \quad & \sum_{v \in V} \sum_{s \in S} \mu_v(s)\phi_v(s) + \sum_{f \in F} \sum_{\{s_v\} \in S^{|f|}} \nu^f(\{s_v\})\psi(\{s_v\}) \\ \text{subject to} \quad & \mu_v(s) = \nu^f(s_v) \;\; \forall f \in F, v \in f, s \in S \\ & \mu_v(s) \geq 0 \;\; \forall v \in V, s \in S \\ & \sum_{s \in S} \mu_v(s) = 1 \;\; \forall v \in V \end{aligned} \tag{10}$$

where $\nu^f(s_v)$ denotes the marginal probability of s_v

$$\nu^f(s_v) = \sum_{\{s_{v'}\}_{v' \in f - \{v\}} \in S^{|f|-1}} \nu^f(\{s_v\} \cup \{s_{v'}\}) \tag{11}$$

Therefore, the linear program has $|S|\sum_{f\in F}|f| + |V||S| + |V|$ constraints.

(i) Derive the dual of this linear program.

Let $a^f_{v,s}$, $b_{v,s}$ and c_v be the dual variables associated with each of the constraints. Then, at any feasible point we have

$$\sum_{f\in F}\sum_{v\in f}\sum_{s\in S} a^f_{v,s}(\mu_{v,s} - \nu^f(s_v)) - \sum_{v\in V}\sum_{s\in S} b_{v,s}\mu_{v,s} + \sum_{v\in V} c_v \sum_{s\in S}\mu_{v,s} \leq \sum_{v\in V} c_v \tag{12}$$

for $b_{v,s} \geq 0$.

Next, note that for any factor f we can rewrite

$$\begin{aligned}\sum_{v\in f}\sum_{s\in S} a^f_{v,s}\nu^f(s_v) &= \sum_{v\in f}\sum_{s\in S} a^f_{v,s} \sum_{\{s_{v'}\}_{v'\in f-\{v\}}\in S^{|f|-1}} \nu^f(\{s_v\}\cup\{s_{v'}\}) \\ &= \sum_{\{s_v\}_{v\in f}\in S^{|f|}} \nu^f(\{s_v\}) \sum_{v'\in f} a^f_{v',s_{v'}}\end{aligned} \tag{13}$$

which we can use to rearrange the left hand side of our inequality

$$\sum_{v\in V}\sum_{s\in S}\mu_{v,s}\left(c_v - b_{v,s} + \sum_{f:v\in f} a^f_{v,s}\right) - \sum_{f\in F}\sum_{\{s_v\}\in S^{|f|}} \nu^f(\{s_v\}) \sum_{v'\in f} a^f_{v',s_{v'}} \leq \sum_{v\in V} c_v \tag{14}$$

giving us the dual problem

$$\begin{aligned}\text{minimize} \quad & \sum_{v\in V} c_v \\ \text{subject to} \quad & \phi_v(s) = c_v - b_{v,s} + \sum_{f:v\in f} a^f_{v,s} \quad \forall v\in V, s\in S \\ & \psi(\{s_v\}_{v\in f}) = -\sum_{v'\in f} a^f_{v',s} \quad \forall f\in F, \forall\{s_v\}_{v\in f}\in S^{|f|}\end{aligned} \tag{15}$$

3 A bad day for the simplex algorithm

In this question, we'll explore the geometry of linear programming and a worst case for the simplex algorithm using a variant of the Klee-Minty cube. Let's define the cube in n dimensions as

$$0 \leq x_k \leq 2^k - 1 - 2\sum_{i=1}^{k-1} x_i. \qquad \forall k \in \{1, 2, \ldots, n\} \tag{16}$$

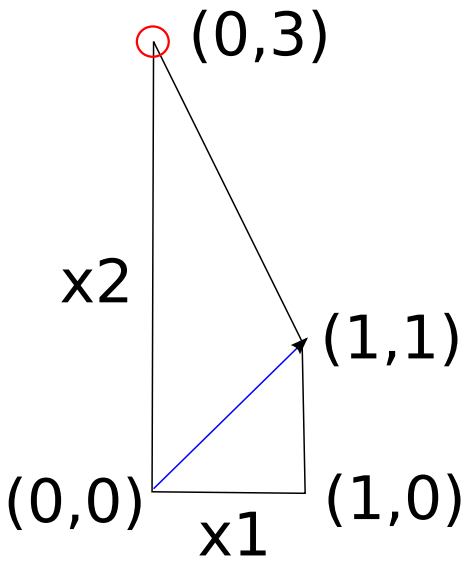
Note that for $k = 1$, the upper bound constraint degenerates to $x_1 \leq 1$.

(a) Sketch or plot the cube in $\mathbb{R}^2$.

Our worst-case example maximizes the sum of the x_i's over this cube.

(b) On your sketch, draw the objective function as a vector and circle the optimal vertex.

Solution for (a) and (b):



Let's run the simplex algorithm on the cube in $\mathbb{R}^2$ and see what it does.

(c) Convert the following linear program into standard form. Name your slack variables y_1 and y_2.

$$\min_x \ x_1 + x_2, \text{ subject to:} \tag{17}$$
$$0 \leq x_1 \leq 1 \tag{18}$$
$$0 \leq x_2 \leq 3 - 2x_1 \tag{19}$$

Solution:

$$\min_x \ x_1 + x_2, \text{ subject to:} \tag{20}$$
$$x_1 + y_1 = 1 \tag{21}$$
$$2x_1 + x_2 + y_2 = 3 \tag{22}$$
$$x_1, x_2, y_1, y_2 \geq 0 \tag{23}$$

(d) Write out the simplex tableau for the point $x_1 = 0, x_2 = 0$ denoting z as the objective value. Which variables are in our starting basis?

Solution:

$$y_1 = 1 - x_1 \tag{24}$$
$$y_2 = 3 - 2x_1 - x_2 \tag{25}$$
$$z = x_1 + x_2 \tag{26}$$

y_1 and y_2 are in our starting basis.

(e) Which variables are legal variables to enter our basis? Why?

Solution:
Both x_1 and x_2 have positive coefficients in the tableau and therefore may enter.

Let's have the simplex algorithm choose the entering variable with the maximum objective coefficient in current tableau breaking ties by picking the variable with the smallest index.

(f) Which variable enters the basis, and which leaves? Write out the new tableau. What are the values of x_1 and x_2? From this point, how many more iterations will be required?

Solution:
x_1 enters and y_1 leaves.

$$x_1 = 1 - y_1 \tag{27}$$
$$y_2 = 1 - x_2 \tag{28}$$
$$z = 1 - y_1 + x_2 \tag{29}$$

In this basis, $x_1 = 1, x_2 = 0$. Two more iterations will be required.

Let's take our intuition of what's happening from the 2-dimensional case and examine the performance of the simplex algorithm in $\mathbb{R}^n$. Here, we will consider the program

$$\min_x \sum_{i=1}^{n} x_i, \text{ subject to:} \tag{30}$$
$$0 \leq x_k \leq 2^k - 1 - 2\sum_{i=1}^{k-1} x_i. \qquad \forall k \in \{1, 2, \ldots, n\} \tag{31}$$

(g) How many vertices, edges and faces does the polytope being optimized over have as a function of n? Here, a face is a hyperplane that touches n points on the boundary (no 3 of which are colinear) that does not intersect the interior.

Solution:
There are 2^n vertices, $n2^n/2$ edges and $2n$ facets. Just like a normal cube.

(h) Prove that any feasible basis includes either x_k, or y_k, but not both. Remember that a basis is a corner point and cannot be written as a linear combination of two other feasible points.

Solution:
First, we show that x_k and y_k cannot both be zero by showing that the unit cube fits inside the Klee-Minty cube. If $0 \leq x \leq 1$, for all k we have

$$0 \leq x_k + 2\sum_{i=1}^{k-1} x_i \tag{32}$$

$$\leq 2(k-1) + 1 = 2k - 1 \tag{33}$$

$$\leq 2^k - 1 \tag{34}$$

Assume there is a basis where $x_k, y_k \geq \epsilon > 0$. Consider the points $x'_k = x_k - \epsilon$, $y'_i = y_i + \epsilon$ and $x''_k = x_k + \epsilon$, $y''_i = y_i - \epsilon$ for $i = \{k, k+1, \ldots, n\}$ with all other coordinates remaining fixed. These two points are feasible and their midpoint is our basis, contradicting that it is a corner.

(i) Show the program has optimum $x^* = (0, 0, \ldots, 0, 2^n - 1)$ by writing out the corresponding tableau.

Solution:
At x^*, we have $y^* = (1, 3, \ldots, 2^{n-1} - 1, 0)$. The associated tableau looks like

$$y_k = 2^k - 1 - 2\sum_{i=1}^{k-1} x_i \qquad \forall k \in \{1, 2, \ldots, n-1\} \tag{35}$$

$$x_n = 2^n - 1 - y_n - 2\sum_{i=1}^{n-1} x_i \tag{36}$$

$$z = 2^n - 1 - y_n - \sum_{i=1}^{n-1} x_i \tag{37}$$

This tableau has no positive coefficients on the non-basis variables in the objective value and is therefore optimal.

Given that x_i and y_i cannot be in a basis together, we know that when x_i enters, then y_i exits and vice versa. From this, we can represent a transformation to a basis with a number between 1 and n. That is, the transformation 2 will have x_2 enter and y_2 exit if y_2 is in the basis.

(j) If $\{a_1, a_2, \ldots, a_m\}$, where for all i, $a_i \in \{1, \ldots, n\}$, is the sequence of entering variables that solves the n dimensional cube, show that the sequence $\{a_1, a_2, \ldots, a_m, n+1, a_1, a_2, \ldots, a_m\}$ solves the $n+1$ dimensional cube.
Hint: Use your tableau from part (i).

Solution:
The tableau after the sequence completes is

$$y_k = 2^k - 1 - 2\sum_{i=1}^{k-1} x_i \qquad \forall k \in \{1, 2, \ldots, n-1\} \tag{38}$$

$$x_n = 2^n - 1 - y_n - 2\sum_{i=1}^{n-1} x_i \tag{39}$$

$$y_{n+1} = 2^{n+1} - 1 - x_{n+1} - 2\sum_{i=1}^{n} x_i \tag{40}$$

$$z = 2^n - 1 - \sum_{i=1}^{n-1} x_i - y_n + x_{n+1}. \tag{41}$$

We have one choice for an entering variable, x_{n+1}, which results in the tableau

$$y_k = 2^k - 1 - 2\sum_{i=1}^{k-1} x_i \qquad \forall k \in \{1, 2, \ldots, n-1\} \tag{42}$$

$$x_n = 2^n - 1 - y_n - 2\sum_{i=1}^{n-1} x_i \tag{43}$$

$$x_{n+1} = 2^{n+1} - 1 - y_{n+1} - 2\sum_{i=1}^{n} x_i \tag{44}$$

$$z = 2^n + \sum_{i=1}^{n-1} x_i + y_n - y_{n+1}. \tag{45}$$

We notice that since no transformations effect $n + 1$ ever again we can remove it from consideration, which results in the starting tableau for the n dimensional problem, except that x_n and y_n have been swapped. This is actually inconsequential, though, as they have the same coefficients everywhere in the tableau once x_{n+1} has been removed.

(k) How many iterations does the simplex algorithm take when starting from $x = (0, 0, \ldots, 0)$ on the n-dimensional Klee-Minty cube? Use induction where your answer from (f) is the base case and your answer from (j) is the inductive step.

Solution: $2^n - 1$.

The case presented here is a contrived worst-case. As stated in class, in practice the simplex algorithm often performs quite well. Smoothed analysis, where one bounds the expected performance of an algorithm on a slightly perturbed worst-case input, is a tool that attempts to describe this phenomenon. See D. Spielman and S. Teng, "Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time", ACM Symposium on Theory of Computing, 2001 for more details.

4 Newton's method

In this problem, we will implement the Newton's method to solve logistic regression. Given a dataset with two types of binary labels (y_i) and and p dimensional features ($\mathbf{x}_i$) ($i = 1, \cdots, n$), we will estimate $\theta \in \mathcal{R}^p$ that minimizes the negative of log likelihood (equivalently maximizing the log likelihood). Our objective function would be

$$-\log(L(\theta|X,Y)) = \frac{1}{n}\sum_{i=1}^{n}(-y_i \log(\sigma_\theta(\mathbf{x}_i)) - (1-y_i)\log(1-\sigma_\theta(\mathbf{x}_i))) \tag{46}$$

where $\sigma_\theta(\mathbf{x}) = \dfrac{1}{1+\exp(-\mathbf{x}^T\theta)}$, $y_i \in \{0,1\}$ refers to i th element of Y, $\mathbf{x}_i \in \mathcal{R}^p$ corresponds to the ith column of X and n is the total number of samples.

Taking the derivative of the objective function with respect to the parameter (θ) results in the gradient,

$$\frac{d(-\log(L(\theta|X,Y)))}{d\theta} = \frac{1}{n}\sum_{i=1}^{n}(\sigma_\theta(\mathbf{x}_i) - y_i)\mathbf{x}_i) \tag{47}$$

Finally, the Hessian of our objective function results in

$$\frac{d^2(-\log(L(\theta|X,Y)))}{d\theta^2} = \frac{1}{n}\sum_{i=1}^{n}\left[\sigma_\theta(\mathbf{x}_i)(1-\sigma_\theta(\mathbf{x}_i))\mathbf{x}_i\mathbf{x}_i^T\right] \tag{48}$$

(a) [2 pts] Implement code for computing the likelihood of the data. Please name it computeL.m or computeL.r. The input to the computeL function should be θ, X and Y. The output should be the objective function value.

SOLUTION: note that you can easily avoid for-loops.

```
function L = computeL(theta,X,Y)

L = mean(-Y'.*log(1./(1+exp(-theta'*X)))-(1-Y').*log(exp(-theta'*X
    )./(1+exp(-theta'*X))));
```

(b) [2 pts] Implement code for computing the gradient of the likelihood with respect to θ. Please name it computeG.m or computeG.r. The input to the computeG function should be θ, X and Y. The output should be the gradient.

SOLUTION: note that you can easily avoid for-loops. You can make your code more efficient by using bsxfun in MATLAB.

```
function g = computeG(theta,X,Y)

g = mean(X*diag((1./(1+exp(-theta'*X))-Y')),2);
```

(c) [2 pts] Implement code for computing the Hessian of the likelihood with respect to θ. Please name it computeH.m or computeH.r. The input to the computeH function should be θ and X. The output should be the value of the Hessian.

SOLUTION:

```
function h = computeH(theta,X)

m = size(X,2);
h = 1/m*(X*diag((sigmoid(-theta'*X))))*(X*diag(1-sigmoid(-theta'*X
    ))))';
```

(d) [5 pts] Implement backtrack line search method (Boyd and Vandenberghe Ch. 9.2). Name the code backtrack.m or backtrack.r. The input to the function should be θ, the pointers to the functions computing the objective function (computeL) and the gradient (computeG), the direction along which you are moving (for the gradient method this will correspond to $-g$ and for the Newton's method this will correspond to $-H^{-1}g$), α, β and the input you need to compute the objective function and the gradient (X and Y). To prevent infinitely reducing the step size until the criterion is met, exit the program when step size is too small (say, when the stepsize is less than 10^{-6}). The output should include the new estimate of θ, the stepsize used, the new objective function value and the number of times computeL was called.

```
function [newtheta, t, newobj,nfc] = backtrack(theta,objfunc,
    gradfunc,delta,alpha,beta,dataparam);

t = 1;
fc = objfunc(theta,dataparam{:});
fn = objfunc(theta+t*delta,dataparam{:});
nfc = 3;
d =  gradfunc(theta,dataparam{:});

while(fn>fc+alpha*t*d'*delta)
    t = beta*t;
    fn = objfunc(theta+t*delta,dataparam{:});
    nfc = nfc + 1;
    if t<1e-6
        break;
    end
end

newtheta = theta+t*delta;
newobj   = objfunc(theta+t*delta,dataparam{:});
```

Please download wine-data.mat from `http://www.cs.cmu.edu/~ggordon/10725-F12/hws/`

`hw3/`. This data includes the quality ratings of 1599 types of wine (1 corresponds to good, 0 to bad) and the volatile acidity (the first row of X) and the alcohol ratio (the second row of X) of each type.

(e) [2 pts] Run the Newton's method and the gradient descent method for 5 iterations with backtracking line search. Start from $\theta_0 = \mathbf{0}$ (e.g., a vector with all elements set to 0). Use $\alpha = 0.01$ and $\beta = 0.8$ for the backtracking line search. For both algorithms, plot the objective function values at each iteration including the initial value of the objective function.

SOLUTION:

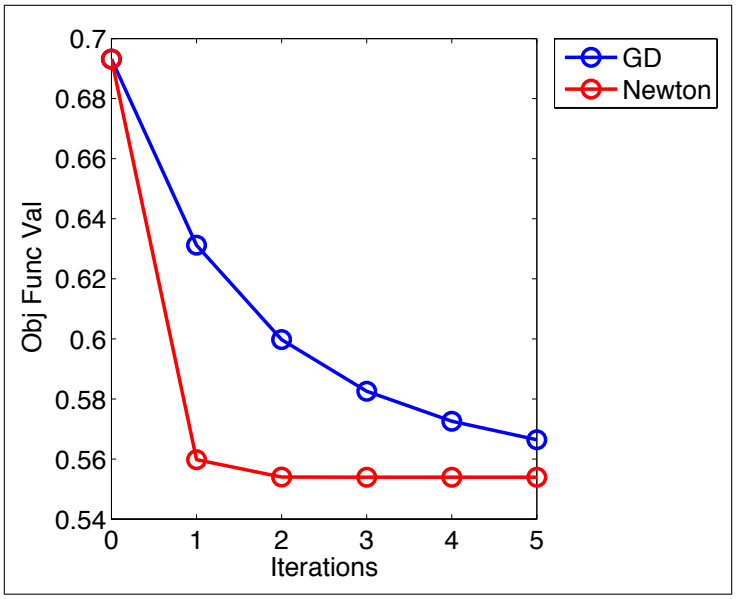


Figure 1: Figure for 4(e)

(f) [2 pts] Plot your data (using distinct markers or color for the two classes) and the separating hyperplane estimated by the Newton's method and the gradient method. How would you conclude on the volatile acidity and alcohol level's relation to the quality of wine?

SOLUTION: Good wines and bad ones are not totally separable, but those with more alcohol and low volitive acidity tends to be categorized as good ones (Figure 2).

In addition to the printout, please submit your code by
1. zipping the three files for (a) - (d) into one zip file (please name it youAndrewID_hw3.zip)
2. send the zip file to submission10725f12@gmail.com before the due date. Please include in the subject line of your email, your name, AndrewID and the assignment number.

Now that we know your code works fine with low dimensional data, let's move onto more

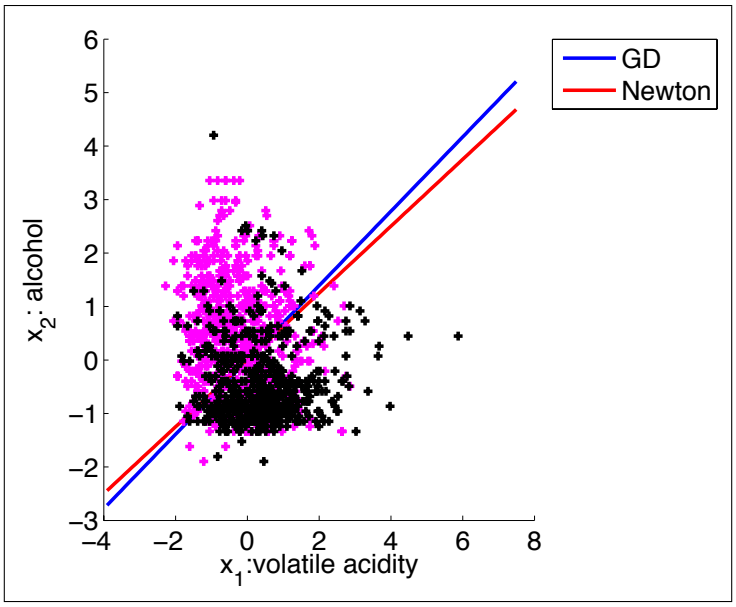


Figure 2: Figure for 4(f)

high dimensional data. Please download newton-data.mat from `http://www.cs.cmu.edu/~ggordon/10725-F12/hws/hw3/`. . The data consists of 4500 samples. The dimensionality of the features is 1000. Run the gradient descent method with backtracking line search for 30 iterations and the Newton's method with backtracking linear search for 10 iterations starting from $\mathbf{0}$.

(g) [2 pts] Plot the objective function values as a function of the number of iterations including the initial objective function values for both the gradient method and the Newton's method in one figure.

SOLUTION: Please refer to Figure 3.

(h) [2 pts] Plot the objective function values as a function of the running time t for both the gradient method and the Newton's method in one figure. Please include the initial objective function value at $t = 0$. Compute the elapsed time per iteration. (Hint: you can use the tic and toc functions in MATLAB to record the elapsed times.)

SOLUTION: Please refer to Figure 4.

(i) [2 pts] Plot the histogram of step sizes returned by the backtracking line search over all iterations for one run. Make separate plots for the gradient method and the Newton's method each. Does the stepsize distribution for the descent gradient method differ from that of the Newton's method? Explain why.

SOLUTION: Please refer to Figure 5.

(j) [2 pts] Plot the histogram of number of times the function that computes the objective

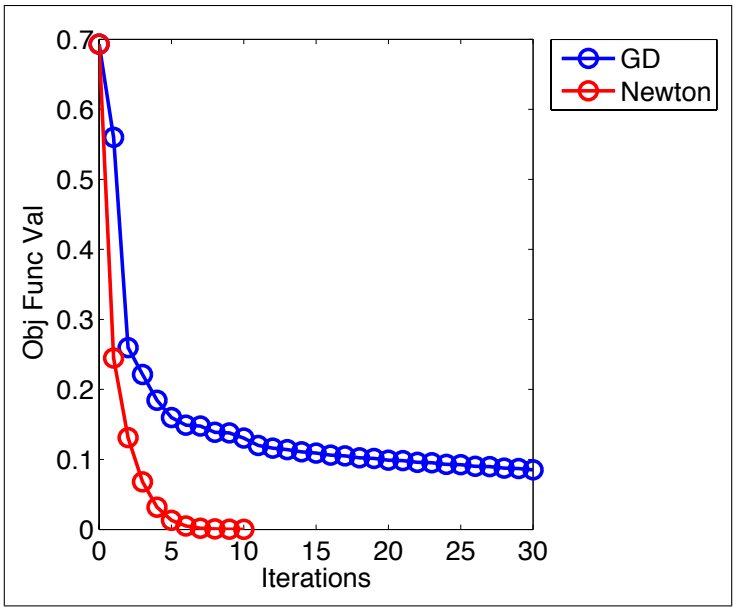


Figure 3: Figure for 4(g)

function was called by the backtracking line search over all iterations. Make separate plots for the gradient method and the Newton's method each. Does the histogram for the descent gradient method differ from that of the Newton's method? Explain why.

SOLUTION: Newton's method always returns step size 1. This is because Please refer to Figure 6.

(k) [2 pts] Run your code again starting from $\theta_0 = \mathbf{1}$. Which algorithm do you think is more sensitive to the initial points? Explain why.

SOLUTION: Newton's method is more sensitive to the initial point. This is because the second order approximation is less accurate when the initial point is far from the optimal point.

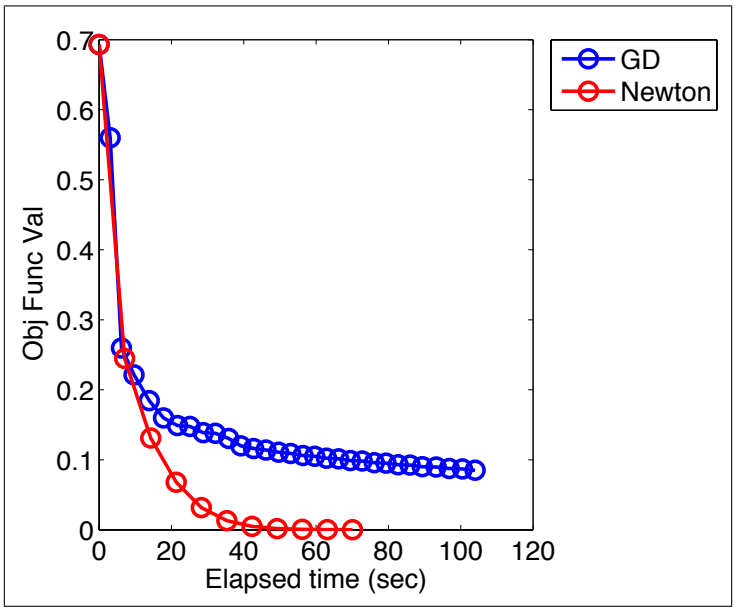


Figure 4: Figure for 4(h)

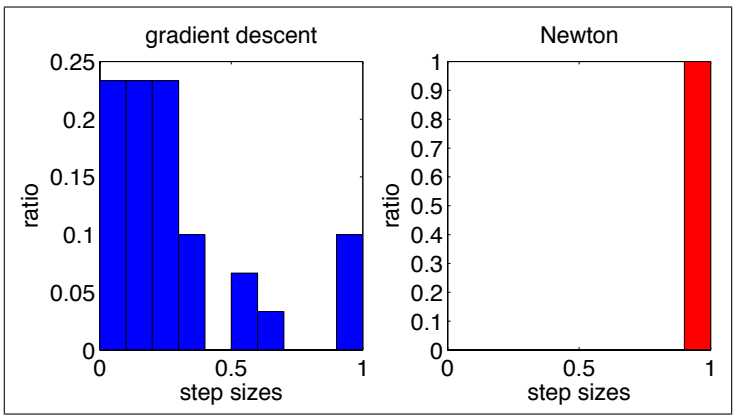


Figure 5: Figure for 4(i)

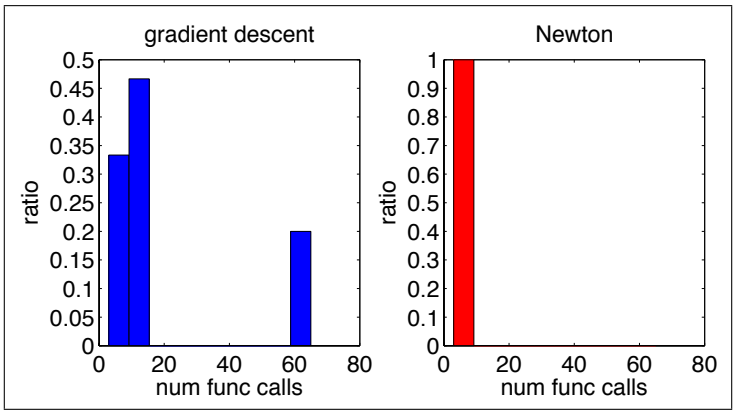


Figure 6: Figure for 4(j)