$\log e : x$

$\boxed{\text{Problem 1 : MLE of Multivariate Gaussian Distribution}}$

$x_1 \dots x_n \in \mathbb{R}^d$

a) $X \sim N(\mu, \sigma^2 I_{d \times d})$.

$$P(x_i) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} e^{-\frac{1}{2}\left((x_i-\mu)^T \Sigma^{-1} (x_i-\mu)\right)}$$

$$L = \sum_{i=1}^{N} \log P(x_i)$$

$$\log P(x_i) = -\frac{d}{2}\log 2\pi - \frac{1}{2}\log(|\Sigma|) - \frac{1}{2}(x_i-\mu)^T \Sigma^{-1}(x_i-\mu)$$

$$L = -\frac{Nd}{2}\log 2\pi - \frac{N}{2}\log|\Sigma| - \sum_{i=1}^{N} \frac{(x_i-\mu)^T \Sigma^{-1}(x_i-\mu)}{2}$$

$$\Sigma = \begin{bmatrix} \sigma^2 & & \\ & \sigma^2 \ddots & \\ & & \sigma^2 \end{bmatrix} \rightarrow |\Sigma| = \underbrace{(\sigma^2)(\sigma^2)\dots(\sigma^2)}_{d} = \sigma^{2d}$$

$$\Sigma^{-1} = \begin{bmatrix} \sigma^{-2} & & \\ & \sigma^{-2} & \\ & & \ddots \sigma^{-2} \end{bmatrix} \rightarrow (x_i-\mu)^T \Sigma^{-1}(x_i-\mu) = \sigma^{-2}(x_i-\mu)^T(x_i-\mu)$$

$$L = \frac{-nd}{2}\log(2\pi) - nd\log\sigma - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^T(x_i-\mu)$$

$$L = nd\left(-\log\sqrt{2\pi} - \log\sigma\right) - \sum_{i=1}^{n} \frac{\|x_i-\mu\|_2^2}{2\sigma^2}$$

$$\frac{\partial L}{\partial \mu} = -\frac{1}{2\sigma^2}(-2)\sum_{i=1}^{n}(x_i-\mu) = 0 \Rightarrow \sum_{i=1}^{n} x_i - \mu = 0$$

$$\boxed{\mu = \frac{\sum x_i}{N}}$$

$$\frac{\partial L}{\partial \sigma} = \frac{-nd}{\sigma} - \frac{1}{\cancel{2}\sigma^3}(-\cancel{2})\sum_{i=1}^{N}(x_i-\mu)^T(x_i-\mu) = 0$$

$$0 = \frac{-nd}{\sigma} + \frac{1}{\sigma^3}\sum_{i=1}^{N}\|x_i-\mu\|_2^2 \Rightarrow \boxed{\sigma^2 = \frac{\sum_{i=1}^{N}\|x_i-\mu\|_2^2}{Nd}}$$

[1 2 3 4 5] [10] $\begin{bmatrix} 3 \\ 4 \\ 5 \\ \end{bmatrix}$ 10

$d \times d$ $d \times 1$

2 b) $X \sim \mathcal{N}(\mu, \Lambda)$ where $\Lambda = \begin{bmatrix} \sigma_1^2 & \sigma_2^2 & \dots & \sigma_d^2 \end{bmatrix}$ $X: x_1 \dots x_N \in \mathbb{R}^d$

$$P(x_i) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} e^{-\frac{1}{2}(x_i - \mu)^T \Sigma^{-1} (x_i - \mu)}$$

$$L = \sum_i^N \log P(x_i)$$

$$\log P(x_i) = -d \log \sqrt{2\pi} - \log \sqrt{|\Sigma|} - \frac{1}{2}(x_i - \mu)^T \Sigma^{-1} (x_i - \mu)$$

$$|\Sigma| = \sigma_1^2 \sigma_2^2 \cdots \sigma_d^2 = \left(\prod_j^d \sigma_j\right)^2 \Rightarrow \sqrt{|\Sigma|} = \prod_j^d \sigma_j$$

$$\log \sqrt{|\Sigma|} = -\sum_j^d \log \sigma_j$$

learn Trace method $\longrightarrow$ $\Sigma^{-1} = \begin{bmatrix} \sigma_1^{-2} & & & \\ & \sigma_2^{-2} & & \\ & & \ddots & \\ & & & \sigma_d^{-2} \end{bmatrix} \Rightarrow \frac{1}{2}(x_i - \mu)^T \Sigma^{-1}(x_i - \mu) = \frac{1}{2}\left[\sigma_1^{-2} \sigma_2^{-2} \cdots \sigma_d^{-2}\right] \|x_i - \mu\|_2^2$

$$= \frac{1}{2} \operatorname{Tr}\left(\Sigma^{-1} \|x_i - \mu\|_2^2\right)$$

$$= \frac{1}{2} \|x_i - \mu\|_2^2 \sum_{j=1}^d \sigma_j^{-2}$$

$$\log P(x_i) = -d \log \sqrt{2\pi} - \sum_j^d \log \sigma_j - \frac{1}{2}\|x_i - \mu\|_2^2 \sum_{j=1}^d \sigma_j^{-2}$$

$$L = -Nd \log \sqrt{2\pi} - N \sum_j^d \log \sigma_j - \sum_j^d \frac{1}{2\sigma^2} \sum_i^N \|x_i - \mu\|_2^2$$

$$\frac{\partial L}{\partial \mu} = \cancel{-} \sum_{j=1}^d \frac{\cancel{+2}}{\cancel{2}\sigma_j^2} \sum_{i=1}^N (x_i - \mu) = 0 = \left(\sigma_1^{-2} + \sigma_2^{-2} + \dots + \sigma_d^{-2}\right) \sum_{i=1}^N (x_i - \mu) = 0$$

$$\boxed{\mu = \frac{\sum^N x_i}{N}}$$

$$\frac{\partial L}{\partial \Lambda} = -N \sum_{j=1}^d \frac{1}{\sigma_j} + \sum_{j=1}^d \frac{\cancel{+2}}{\cancel{2}\sigma_j^3} \sum_{i=1}^N \|x_i - \mu\|_2^2 = 0$$

$$N\left(\sigma_1^{-1} + \sigma_2^{-1} + \dots + \sigma_d^{-1}\right) = \left(\sigma_1^{-3} + \sigma_2^{-3} + \dots + \sigma_d^{-3}\right) \sum_{i=1}^N \|x_i - \mu\|_2^2$$

$$\frac{\sum_j^d \frac{1}{\sigma_j}}{\sum_j^d \left(\frac{1}{\sigma_j}\right)^3} = \frac{\sum_i^N \|x_i - \mu\|_2^2}{N}$$

$X \sim N(\mu, \Lambda)$ where $\Lambda = \begin{bmatrix} \sigma_1^2 & & \\ & \sigma_2^2 & \\ & & \ddots \; \sigma_d^2 \end{bmatrix}$

$$\log P(x_i) = -d \log\sqrt{2\pi} - \log\sqrt{|\Lambda|} - \frac{1}{2}(x_i-\mu)^T \Lambda^{-1} (x_i-\mu)$$

$$|\Lambda| = \text{Tr}\,\Lambda \Rightarrow -\log\sqrt{|\Lambda|} = -\frac{1}{2}\log \text{Tr}\,\Lambda$$

$$L = \sum^N \log P(x_i) = -Nd\log\sqrt{2\pi} - \frac{N}{2}\log(\text{Tr}\,\Lambda) - \sum^N \frac{(x_i-\mu)^T\Lambda^{-1}(x_i-\mu)}{2}$$

$$\frac{\partial L}{\partial \mu} = + \sum_{i=1}^{N} \frac{+\cancel{2}\Lambda^{-1}(x_i-\mu)}{\cancel{2}} = 0 \Rightarrow \boxed{\mu = \frac{\sum^N x_i}{N}}$$

$$\frac{\partial L}{\partial \Lambda} = \frac{-N}{\cancel{2}} \frac{1}{\text{Tr}\,\Lambda} \frac{d\,\text{Tr}\,\Lambda}{d\Lambda} - \frac{1}{\cancel{2}} \sum_{i=1}^{N} \frac{\partial}{\partial \Lambda} (x_i-\mu)^T \Lambda^{-1} (x_i-\mu) = 0$$

$$L = -Nd\log\sqrt{2\pi} - \frac{N}{2}\log|\Lambda| - \frac{1}{2}\sum_1 (x_i-\mu)^T\Lambda^{-1}(x_i-\mu)$$

$$\frac{\partial L}{\partial \Lambda} = -\frac{N}{2}\frac{1}{|\Lambda|}\frac{\partial|\Lambda|}{\partial\Lambda} - \frac{1}{2}\sum_1^N \frac{\partial}{\partial\Lambda}(x_i-\mu)^T\Lambda^{-1}(x_i-\mu)$$

$$\frac{\partial|\Lambda|}{\partial\Lambda} = |\Lambda|(\Lambda^{-1})^T = |\Lambda|(\Lambda^T)^{-1} = |\Lambda|\,\Lambda^{-1}$$

$$\frac{\partial L}{\partial \Lambda} = -\frac{N}{2}(\Lambda^{-1}) - \frac{1}{2}\sum_{i=1}^N \frac{\partial}{\partial\Lambda}\underbrace{(x_i-\mu)^T\Lambda^{-1}(x_i-\mu)}_{a} = 0$$

$$\Lambda\Lambda^{-1} = I \Rightarrow \frac{\partial}{\partial\Lambda}(\Lambda\Lambda^{-1}) = 0 = \left(\frac{\partial}{\partial\Lambda}\Lambda\right)\Lambda^{-1} + \Lambda\left(\frac{\partial}{\partial\Lambda}\Lambda^{-1}\right) = 0$$

$$\frac{\partial}{\partial\Lambda}\Lambda^{-1} = -\Lambda^{-1}\left(\frac{\partial}{\partial\Lambda}\Lambda\right)\Lambda^{-1} = -\Lambda^{-1}\Lambda^{-1}$$

b/c Λ^{-1} diagonal $\Rightarrow \frac{\partial}{\partial\Lambda}\Lambda^{-1} = -\Lambda^{-2}$

(?)

$$\frac{\partial L}{\partial\Lambda} = 0 = \frac{-N\Lambda^{-1}}{\cancel{2}} + \frac{1}{\cancel{2}}\sum_{i=1}^N + \|x_i-\mu\|_2^2\,\Lambda^{-2} = 0$$

$$\boxed{\Lambda = \frac{\sum_{i=1}^N \|x_i-\mu\|_2^2}{N}}$$

c). $X \sim \mathcal{N}(A\mu, \Lambda) \quad A, D \in \mathbb{R}^{d \times d} \quad \mu = ?$

$$P(x_i) = \frac{1}{\sqrt{(2\pi)^d |\Lambda|}} e^{-\frac{1}{2}(x_i - A\mu)^T \Lambda^{-1} (x_i - A\mu)}$$

$$\log P(x_i) = -d \log \sqrt{2\pi} - \log \sqrt{|\Lambda|} - \frac{1}{2}(x - A\mu)^T \Lambda^{-1} (x_i - A\mu)$$

$$L = \sum \log P(x_i) = -Nd \log \sqrt{2\pi} - N \log \sqrt{|\Lambda|} - \frac{N}{2} \sum^N (x_i - A\mu)^T \Lambda^{-1} (x_i - A\mu)$$

$$\frac{\partial L}{\partial \mu} = \frac{N}{2} \sum_{i=1}^N \frac{\partial}{\partial \mu} (x_i - A\mu)^T \Lambda^{-1} (x_i - A\mu) = 0$$

$$= \frac{\cancel{N}}{\cancel{2}} \sum_{i=1}^N - \cancel{2} A \Lambda^{-1} (x_i - A\mu) = 0$$

$$= \cancel{A\Lambda^{-1}} \sum_{i=1}^N (x_i - A\mu) = 0$$

$$\boxed{\mu = \frac{\sum_{i=1}^N x_i}{nA}}$$

$\sqrt{(x+\mu)^T(x+\mu)} = \|x-\mu\|$

Problem 2: l_2-regularized Logistic/Linear Regression w/ Newton's Method

$S: \{(x_i, y_i)\}_{i=1}^n$ training set $x_i \in \mathbb{R}$, $y_i \in \{0,1\}$

a). Negative log-likelihood for l_2-regularized logistic regression:

$$l(\beta) = \lambda \|\beta\|_2^2 - \sum_{i=1}^n \left[y_i \log \mu_i + (1-y_i)\log(1-\mu_i)\right]$$

$\mu_i = \frac{1}{1+e^{-\beta x_i}}$ $\lambda > 0$ is regularization parameter

b). l_2-regularized quadratic cost function (i.e. ridge regression)

$$J(\beta) = \lambda \|\beta\|_2^2 + \frac{1}{2}\sum_{i=1}^N (y_i - \beta x_i)^2$$

$$X = \begin{bmatrix} 4 & 1 \\ 5 & 1 \\ 5.0 & 1 \\ \vdots & \vdots \\ 4.3 & 1 \end{bmatrix} \begin{matrix} \leftarrow x_1 \\ \leftarrow x_2 \\ \\ \\ \leftarrow x_n \end{matrix} \quad n\times 2 \qquad \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} \quad 2\times 1 \qquad Y = \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \leftarrow y_1 \\ \leftarrow y_2 \\ \\ \\ \leftarrow y_n \end{matrix} \qquad M = \begin{bmatrix} \frac{1}{1+e^{-\beta x_1}} \\ \frac{1}{1+e^{-\beta x_2}} \\ \cdots \\ \frac{1}{1+e^{-\beta x_n}} \end{bmatrix}$$

$1\times n \; n\times 2 = 1\times 2$
$2\times n \; n\times 1 = 2\times 1$

1. $\nabla_\beta l(\beta) = ?$, $\nabla_\beta J(\beta) = ?$

$$l(\beta) = \lambda\|\beta\|_2^2 - Y^T \log M + (1-Y^T)\log(1-M)$$

$$l(\beta) = \lambda\|\beta\|_2^2 - Y^T \log\left(1+e^{-X\beta}\right)^{-1} + (1-Y^T)\log\left(1-(1+e$$

$$l(\beta) = \lambda\|\beta\|_2^2 + Y^T \log(1+e^{-X\beta}) + (1-Y^T)\log((2+e^{-X\beta})(1+e^{-X\beta})$$

$$= \lambda\|\beta\|^2 + Y^T\log(1+e^{-X\beta}) + (1-Y^T)\log(2+e^{-X\beta}) - (1-Y^T)\log(1+e$$

$$= \lambda\|\beta\|^2 + (2Y^T-1)\log(1+e^{-X\beta}) + (1-Y^T)\log(2+e^{-X\beta})$$

$$\nabla_\beta l(\beta) = 2\lambda\beta - \frac{2Y^T-1}{1+e^{-X\beta}}(-Xe^{-X\beta}) + \frac{1-Y^T}{2+e^{-X\beta}}(-Xe^{-X\beta}) =$$

1. $\nabla_\beta l(\beta) = ?$, $\nabla_\beta J(\beta) = ?$

$$\nabla_\beta l(\beta) = 2\lambda\beta - \sum_{i=1}^n \frac{y_i}{\mu_i}\frac{\partial \mu_i}{\partial\beta} + \frac{1-y_i}{-\mu_i}\frac{\partial\mu_i}{\partial\beta} = 0$$

$$\frac{d\mu_i}{d\beta} = \frac{\partial}{\partial\beta}\left(1+e^{-\beta x_i}\right)^{-1} = -(1+e^{-\beta x_i})^{-2}(-x_i e^{-\beta x_i}) = \frac{x_i e^{-\beta x_i}}{(1+e^{-\beta x_i})^2}$$

$$\nabla_\beta l(\beta) = 0 = 2\lambda\beta - \sum_{i=1}^n \frac{y_i}{\mu_i}\mu_i^2 x_i e^{-\beta x_i} - \frac{1-y_i}{\mu_i}\mu_i^2 x_i e^{-\beta x_i}$$

$$0 = 2\lambda\beta - \sum_{i=1}^n \frac{(2y_i-1)x_i e^{-\beta x_i}}{1+e^{-\beta x_i}} \Rightarrow 2\lambda\beta - \frac{(2Y^T-1)Xe^{-X\beta}}{(1+e^{-X\beta})} = 0$$

1. Find $\nabla_\beta l(\beta)$, $\nabla_\beta J(\beta)$

a) $\nabla_\beta l(\beta) = 2\lambda\beta - \sum_{i=1}^{N} \frac{y_i}{\mu_i}\frac{\partial \mu_i}{\partial\beta} - \frac{1-y_i}{1-\mu_i}\frac{\partial\mu_i}{\partial\beta} = 0$

$$\frac{\partial\mu_i}{\partial\beta} = \frac{-1}{(1+e^{-x_i\beta})^2}(-x e^{-x_i\beta}) = \frac{1}{1+e^{-x\beta}}\frac{e^{-x_i\beta}}{1+e^{-x_i\beta}} x_i$$

$$= \mu_i(1-\mu_i)x_i$$

$\nabla_\beta l(\beta) = 0 = 2\lambda\beta - \sum_{i=1}^{N}\left(\frac{y_i}{\mu_i} - \frac{1-y_i}{1-\mu_i}\right)\mu_i(1-\mu_i)x_i$

$$2\lambda\beta = \sum_{i=1}^{N}\left(y_i(1-\mu_i) - (1-y_i)\mu_i\right)x_i$$

$$= \sum(y_i - \mu_i y_i - \mu_i + \mu_i y_i)x_i$$

$$2\lambda\beta = \sum_{1}^{N}(y_i - \mu_i)x_i$$

L2-logistic

$$\beta = \frac{\sum(y_i-\mu_i)x_i}{2\lambda} = \boxed{\frac{X^T(Y-M)}{2\lambda} = \beta}$$

$$\boxed{\nabla_\beta l(\beta) = 2\lambda\beta - X^T(Y-M)}$$

b) $J(\beta) = \lambda\|\beta\|_2^2 + \frac{1}{2}(Y-X\beta)^T(Y-X\beta)$

$$\nabla J(\beta) = \lambda\|\beta\|_2^2 + \frac{1}{2}(Y^T - \beta^T X^T)(Y - X\beta)$$

$$J(\beta) = \lambda\|\beta\|_2^2 + \frac{1}{2}(Y^TY - 2Y^TX\beta + \beta^TX^TX\beta)$$

$$\nabla_\beta J(\beta) = 0 = 2\lambda\beta + \frac{1}{2}(-2Y^TX + 2X^TX\beta)$$

$$Y^TX = 2\lambda\beta + X^TX\beta$$

$$\boxed{\beta = (X^TX + 2\lambda I)^{-1}Y^TX} \qquad \boxed{\nabla_\beta J(\beta) = (X^TX + 2\lambda I)\beta - Y^TX}$$

~~2. a) $\nabla_\beta^2 l(\beta) = \frac{\partial}{\partial\beta}\nabla_\beta l(\beta) = \frac{\partial}{\partial\beta}(2\lambda\beta - X^T(Y-M)) = \boxed{2\lambda + X^T} (\mu)(1-\mu) X$~~

~~b) $\nabla_\beta^2 J(\beta) = \frac{\partial}{\partial\beta}\nabla_\beta J(\beta) = \frac{\partial}{\partial\beta}(2\lambda\beta + X^TX\beta - Y^TX)$~~

~~$$= \boxed{2\lambda I + X^TX}$$~~

3. $\ell(\beta) \to \beta^{+} \leftarrow \beta - \dfrac{\nabla_\beta \ell(\beta)}{\nabla_\beta^2 \ell^2(\beta)} = \beta - \dfrac{2\lambda\beta - X^T(Y-\mu)}{2\lambda}$

$$\boxed{\beta_{\ell+1} \leftarrow \beta_\ell - \frac{X^T(Y-\mu)}{2\lambda}}$$

$J(\beta) \to \beta_J \leftarrow \beta_J + \dfrac{\nabla_\beta J(\beta)}{\nabla_\beta^2 J^2(\beta)} = \beta_J - \dfrac{2\lambda\beta + X^T X\beta - Y^T X}{2\lambda I + X^T X}$

$$\boxed{\beta_{J+1} \leftarrow \beta_J - \frac{Y^T X}{(X^T X + 2\lambda I)^2}}$$

$\mu_1(1-\mu_1)$
$\mu_2(1-\mu_2)$

$x_1 \quad x_2$

2. a) $\nabla_\beta^2 \ell(\beta) = \frac{\partial}{\partial \beta} \nabla_\beta \ell(\beta) = \frac{\partial}{\partial\beta}\left(2\lambda\beta - X^T(Y-\mu)\right) = 2\lambda + \frac{\partial}{\partial\beta}(X^T\mu)$

$= 2\lambda + X^T X^T \mu^T (1-\mu)$

$\nabla_\beta^2 J(\beta) = \frac{\partial}{\partial\beta}(X^T X + 2\lambda I)\beta - Y^T X = X^T X + 2\lambda I$

3. $\ell(\beta) \to \beta^{+} \leftarrow \beta - \dfrac{\nabla_\beta \ell(\beta)}{\nabla_\beta^2 \ell^2(\beta)} = \boxed{\beta - \dfrac{\left(2\lambda\beta - X^T(Y-\mu)\right)}{2\lambda I + (X^T)^2 \mu^T (1-\mu)}}$

$J(\beta) \to \beta^{+} \leftarrow \beta - \dfrac{\nabla_\beta J(\beta)}{\nabla_\beta^2 J^2(\beta)} = \boxed{\beta - \dfrac{(X^T X + 2\lambda I)\beta - Y^T X}{2\lambda I + X^T X}}$

$$[\beta_1 \beta_2 \cdots \beta_d]\begin{bmatrix} x_1^T y \\ x_2^T y \\ \vdots \\ x_d^T y \end{bmatrix} \qquad X^T = {}_d\begin{bmatrix} — x_1 — \\ — x_2^T — \\ \vdots \\ — x_d^T — \end{bmatrix} \qquad X = {}_n\begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_d \\ | & | & & | \end{bmatrix}_d$$

$\boxed{\text{Problem 3: } l_1\text{-Regularized Linear Regression}}$

$$X \triangleq n\begin{bmatrix} — x_1 — \\ — x_2 — \\ \vdots \\ — x_n — \end{bmatrix}_d \qquad y \triangleq n\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \qquad \beta \triangleq d\begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_d \end{bmatrix}$$

Lasso reg

Want $y = X\beta \quad \beta = \vec{\beta}$

$X^T X = nI$

Lasso Regression

$$\beta^* = \underset{\beta}{\arg\min}\{J_\lambda(\beta)\} = \underset{\beta}{\arg\min}\left\{\tfrac{1}{2}\|\vec{y} - X\vec{\beta}\|_2^2 + \lambda\|\vec{\beta}\|_1\right\}$$

1. Show β^* determined by i^{th} feature outputs.

Write $J_\lambda(\vec{\beta}) = g(\vec{y}) + \sum_{i=1}^{d} f(X_i, \vec{y}, \beta_i, \lambda)$ where $X_i \triangleq i^{th}$ column of

$$J_\lambda(\beta) = \frac{1}{2}(y - X\beta)^T(y - X\beta) + \lambda\sum_{i=1}^{d}|\beta_i|$$

$$= \frac{1}{2}(y^T - \beta^T X^T)(y - X\beta) + \lambda\sum_{i=1}^{d}|\beta_i|$$

$$= \frac{1}{2}(y^T y - 2\beta^T X^T y + \beta^T X^T X\beta) + \lambda\sum_{i=1}^{d}|\beta_i|$$

$$= \frac{1}{2}y^T y - \sum_{i=1}^{d}\beta_i X_i^T y + \frac{1}{2}\beta^T nI\beta + \lambda\sum_{i=1}^{d}|\beta_i|$$

?

$$= \frac{1}{2}y^T y + \frac{n}{2}\beta^T\beta + \lambda\sum_{i=1}^{d}|\beta_i| - \beta_i X_i^T y$$

$$\boxed{J_\lambda(\beta) = \frac{1}{2}(y^T y) + \frac{\lambda n}{2}\sum_{i=1}^{d}\beta_i^2 + |\beta_i| - \beta_i \vec{X}_i^T\vec{y}}$$

$$g(\vec{y}) = \frac{1}{2}y^T y$$

$$f(X_i, \vec{y}, \beta_i, \lambda) = \frac{\lambda n}{2}\left(\beta_i^2 + |\beta_i| - \beta_i X_i^T\vec{y}\right)$$

2. Assume $\beta_i^* > 0$ ~~$\forall\lambda$~~ $J_\lambda(\beta) > 0$

$$\frac{1}{2}(y^T y) + \frac{\lambda n}{2}\sum_{i=1}^{d}\beta_i^{*2} + |\beta_i^*| - \beta_i^* X_i^T\vec{y} > 0$$

$$J_\lambda(\beta) = \frac{1}{2}(y^T y) + \frac{\lambda n}{2}\sum_{i=1}^{d}\beta_i(\beta_i + 1 - X_i^T\vec{y})$$

$$\frac{\partial J(\beta)}{\partial\beta_i} = \frac{\lambda n}{2}(2\beta_i + 1 - x_i^T y) = 0$$

$$\boxed{\beta_i^* = \frac{x_i^T y - 1}{2}}$$

3. Assume $\beta_i^* < 0$

$$J_\lambda(\beta) = \frac{1}{2} y^T y + \frac{\lambda n}{2} \sum_{i=1}^{d} \beta_i^2 - \beta_i - \beta_i x_i^T y$$

$$\frac{\partial J(\beta)}{\partial \beta_i} = \frac{\lambda n}{2}\left(2\beta_i - 1 - x_i^T y\right) = 0$$

$$\boxed{\beta_i^* = \frac{x_i^T y + 1}{2}}$$

? 4. When $\beta_i^* = 0$

$$\frac{x_i^T y + 1}{2} = 0 \Rightarrow x_i^T y = -1$$

$$\Rightarrow \boxed{|x_i^T y| = 1}$$

$$\frac{x_i^T y - 1}{2} = 0 \Rightarrow x_i^T y = 1$$

5. lasso: $J_\lambda(\beta) = \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1$

Ridge: $J_\lambda(\beta) = \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2$

$$J_\lambda(\beta) = \frac{1}{2} y^T y + \sum_{i=1}^{d} \frac{\lambda n}{2}\left(\beta_i^2 + \beta_i^2 - \beta_i x^T y\right)$$

$$= \frac{1}{2} y^T y + \sum_{i=1}^{d} \frac{\lambda n}{2}\left(2\beta_i^2 - \beta_i x^T y\right)$$

$$\frac{\partial J_\lambda(\beta)}{\partial \beta_i} = 0 = \frac{\lambda \cancel{n}}{\cancel{2}}\left(4\beta_i - x^T y\right)$$

$$\boxed{\beta_i^* = \frac{x_i^T y}{4}} \Rightarrow \boxed{\beta_i^* = 0 \text{ when } x_i^T y = 0}$$

$Y \log M + (1-Y)\log(M)$

$\boxed{\text{Problem 4: Spam classification using logistic Regression}}$

1. Derive gradient descent equations for logistic regression w/ l2 regularization

$p(Y|X,\theta) = \prod_{i=1}^{n} \mu_i^{y_i}(1-\mu_i)^{(1-y_i)}$

$$l(\beta) = \lambda \|\beta\|_2^2 - \sum_{i=1}^{n} y_i \log \mu_i + (1-y_i)\log(1-\mu_i)$$

where $\mu_i = \dfrac{1}{1+e^{-\beta x_i}}$ $\quad \lambda > 0$

$$\nabla_\beta l(\beta) = 2\lambda\beta - \sum_{i=1}^{n} \frac{y_i}{\mu_i}\frac{\partial \mu_i}{\partial \beta} - \frac{1-y_i}{1-\mu_i}\frac{\partial \mu_i}{\partial \beta}$$

$$\frac{\partial \mu_i}{\partial \beta} = \frac{1}{(1+e^{-\beta x_i})^2} x_i e^{-\beta x_i} = \frac{1}{(1+e^{-\beta x_i})} \frac{e^{-\beta x_i}}{1+e^{-\beta x_i}} x_i$$

$$\frac{\partial \mu_i}{\partial \beta} = \mu_i(1-\mu_i)x_i$$

$$\nabla_\beta l(\beta) = 2\lambda\beta - \sum_{i=1}^{n}\left(\frac{y_i}{\mu_i} - \frac{1-y_i}{1-\mu_i}\right)\mu_i(1-\mu_i)x_i$$

$$= 2\lambda\beta - \sum_{i=1}^{n}\left[y_i - y_i\mu_i - (\mu_i - y_i\mu_i)\right]x_i$$

$$= 2\lambda\beta - \sum_{i=1}^{n}(y_i - \mu_i)x_i$$

$$\boxed{\nabla_\beta l(\beta) = 2\lambda\beta - X^T(y-\mu)}$$

$$\beta_{t+1} = \beta_t - \alpha(2\lambda\beta - X^T(y-\mu)) = \beta_t - \alpha\sum_{i=1}^{n} 2\lambda\beta - (y_i - \mu_i)x_i$$

$\boxed{\text{see hw3.py}}$

2. $\beta_{t+1} = \beta_t - \alpha(2\lambda\beta_t - x_i(y_i - \mu_i))$

3. Decreasing alpha over time will allow it to settle into a better optimal solution, so $\boxed{\text{Yes}}$

4. See Kaggle

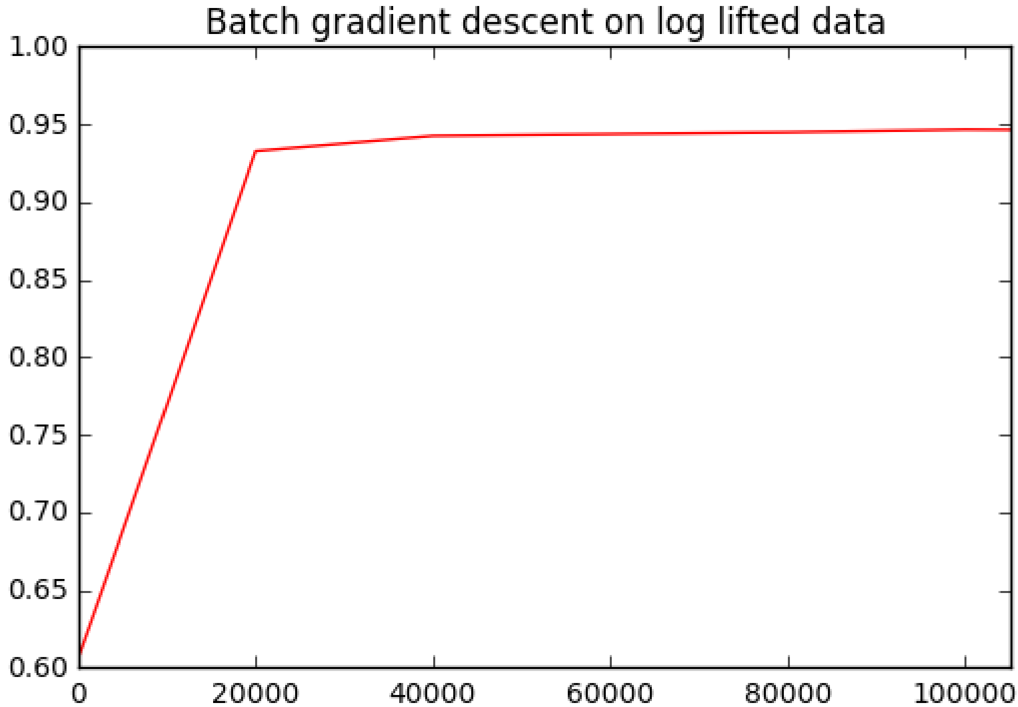
In [1]:

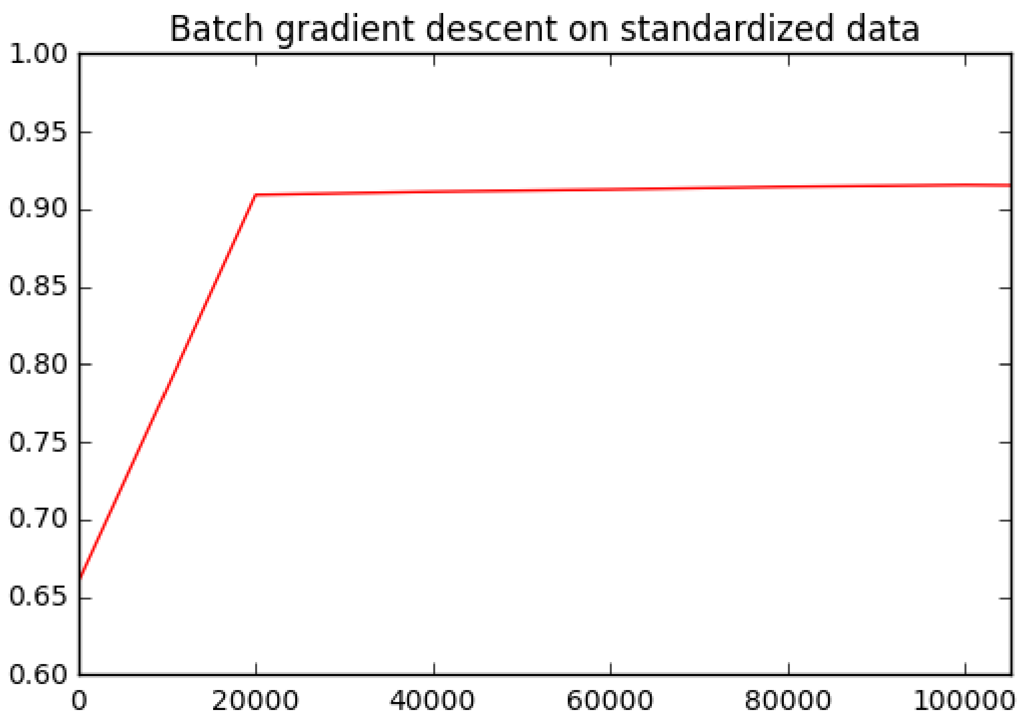
```
import sklearn.metrics as metrics
import numpy as np
import scipy
import scipy.io
import matplotlib.pyplot as plt
import random
import csv
import pdb

%matplotlib inline

clamped = 0.000001
iterations = np.arange(0, 200000, 20000)

def load_dataset():
    mat = scipy.io.loadmat('spam')
    return mat['Xtrain'], mat['Xtest'], mat['ytrain']

def train_gd(X_train, y_train, alpha=1e-7, reg=0.1, num_iter=10000):
    ''' Build a model from X_train -> y_train using batch gradient desce
nt '''
    W = np.zeros((X_train.shape[1], 1))+clamped
    for i in range(num_iter):
        mu = scipy.special.expit(X_train.dot(W))
        gradient = 2*reg*W - X_train.T.dot(y_train - mu)
        W -= alpha * gradient
    return W

def train_sgd(X_train, y_train, alpha=1e-4, reg=0, num_iter=10000):
    ''' Build a model from X_train -> y_train using stochastic gradient
 descent '''
    W = np.zeros((X_train.shape[1], 1))+clamped
    for i in range(num_iter):
        sample_index = random.randint(0, X_train.shape[0] - 1)
        x_i, y_i = X_train[sample_index].reshape(X_train.shape[1], 1), y
_train[sample_index]
        mu = scipy.special.expit(W.T.dot(x_i))
        gradient = 2*reg*W - x_i.dot(y_i - mu)
        W -= alpha * gradient
    return W

def train_sgd_decay(X_train, y_train, alpha=1e-4, reg=0,
num_iter=10000):
    ''' Build a model from X_train -> y_train using stochastic gradient
 descent '''
    W = np.zeros((X_train.shape[1], 1))+clamped
    for i in range(num_iter):
        sample_index = random.randint(0, X_train.shape[0] - 1)
        x_i, y_i = X_train[sample_index].reshape(X_train.shape[1], 1), y
_train[sample_index]
        mu = scipy.special.expit(W.T.dot(x_i))
        gradient = 2*reg*W - x_i.dot(y_i - mu)
        W -= (alpha/(i+1)) * gradient
    return W

def predict(model, X):
    ''' From model and data points, output prediction vectors '''
```

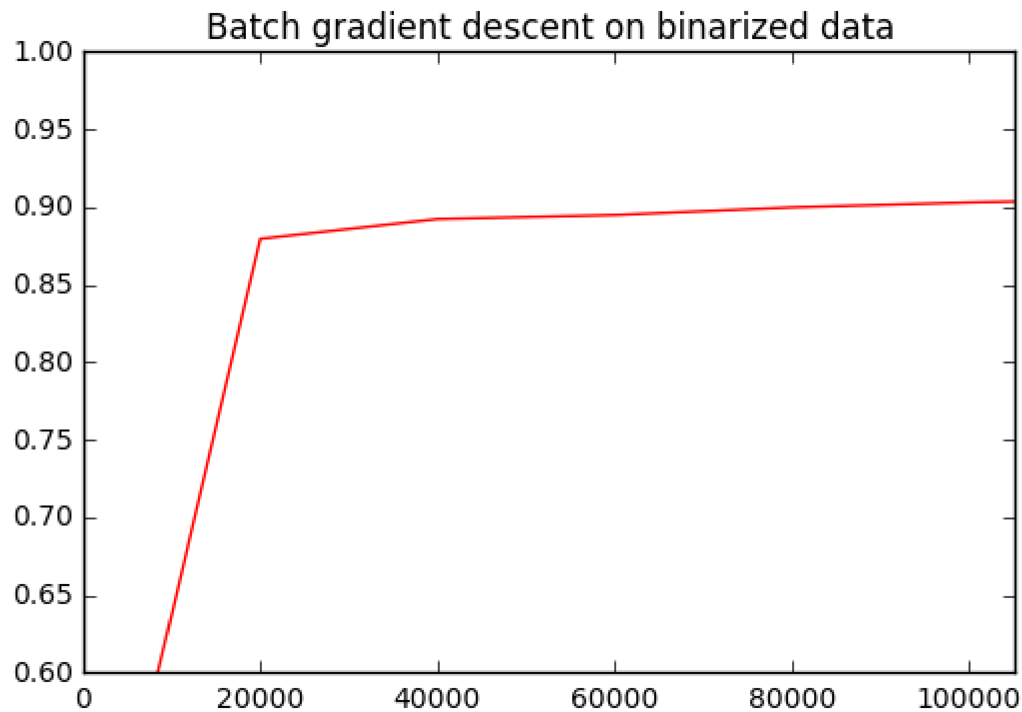
```
    y_pred = scipy.special.expit(X.dot(model))
    return np.round(y_pred)

### Preprocessing Techniques ###
def standardize(X):
    ''' Standardize columns to have mean 0 and unit variance '''
    return (X - np.mean(X, axis=0)) / np.std(X, axis=0)

def log_lift(X):
    ''' Transform the features using log lift '''
    return np.log(X + 0.01)

def binarize(X):
    ''' Binarize features into negative (0) and nonnegative (1) '''
    return np.where(X>0.0, 1.0, 0.0)

def phi(X, G, b):
    ''' Featurize the inputs using random Fourier features '''
    return np.sqrt(2.0/P) * np.cos(np.add(G.dot(X.T), b).T)

def gaussian_random_lift(d):
    mean = np.zeros(d)
    cov = VARIANCE * np.identity(d)
    G = np.random.multivariate_normal(mean, cov, P)
    b = np.random.uniform(0.0, 2 * np.pi, P).reshape((P, 1))
    return G, b


### Post processing analysis ###
def training_loss(X_train, y_train, trainer, alpha=1e-7, reg=0.1):
    training_loss = []
    for num_iter in iterations:
        model = trainer(X_train, y_train, alpha, reg, num_iter)
        pred_y_train = predict(model, X_train)
        accuracy = metrics.accuracy_score(y_train, pred_y_train)
        print("Train accuracy with " + str(num_iter) + " iterations:
 {0}".format(accuracy))
        training_loss.append(accuracy)
    return training_loss

def plot_training_loss(training_loss, title):
    plt.plot(iterations, training_loss, 'r-')
    plt.axis([0, 105000, .6, 1])
    plt.title(title)
    plt.show()

if __name__ == "__main__":
    X_train, X_test, y_train = load_dataset()

    # Uncomment the training_loss calls if you want to see how data was
 fitted

    """ Training losses for gradient descenet """
    # print("Batch gradient descent on log lifted data")
    # log_training_loss = training_loss(log_lift(X_train), y_train, trai
n_gd)
    log = [0.60521739130434782, 0.93275362318840582,
```

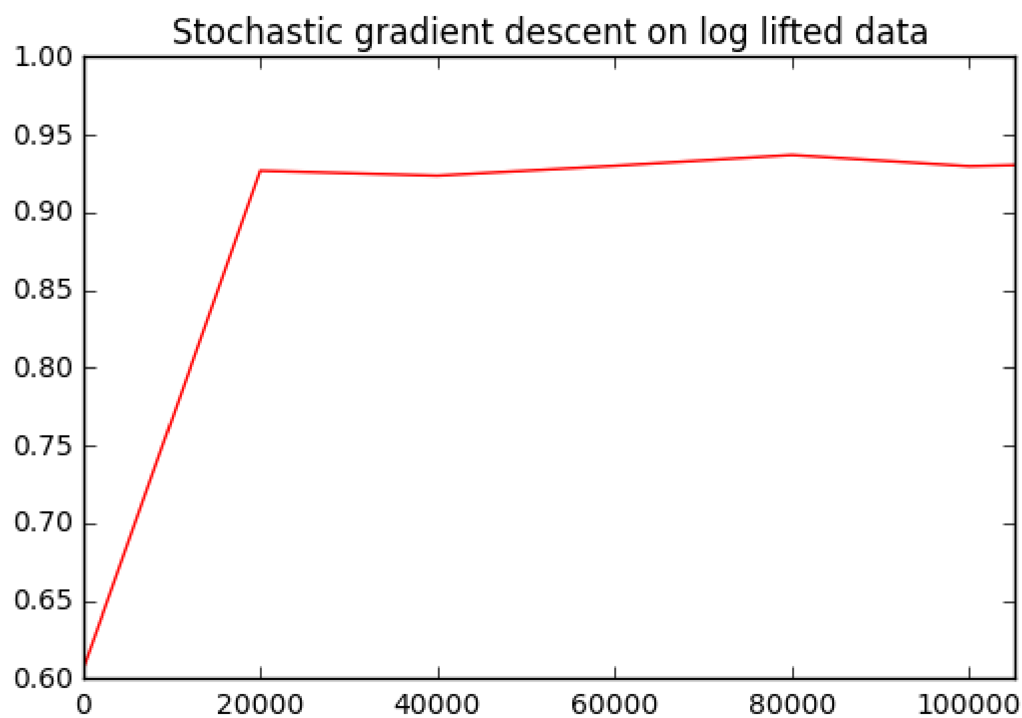
```
0.94260869565217387, 0.94376811594202903, 0.94492753623188408, 0.9466666
6666666666, 0.94608695652173913, 0.94608695652173913, 0.9466666666666666
6, 0.94608695652173913]
    # print("Batch gradient descent on standardized data")
    # standardized_training_loss = training_loss(standardize(X_train), y
_train, train_gd)
    standardized = [0.65884057971014498, 0.90927536231884054, 0.91130434
782608694, 0.9127536231884058, 0.91449275362318838, 0.91565217391304343,
 0.9147826086956522, 0.91594202898550725, 0.91594202898550725, 0.9162318
8405797107]
    # print("Batch gradient descent on binarized data")
    # binarized_training_loss = training_loss(binarize(X_train), y_trai
n, train_gd)
    binarized = [0.39478260869565218, 0.87971014492753619, 0.89246376811
594208, 0.89507246376811589, 0.90000000000000002, 0.90318840579710147,
0.90550724637681157, 0.90666666666666662, 0.91188405797101446, 0.9124637
6811594199]

    """ Training losses for stochastic gradient descenet """
    # print("Stochastic gradient descent on log lifted data")
    # sgd_log_training_loss = training_loss(log_lift(X_train), y_train,
 train_sgd, alpha=1e-4)
    sgd_log = [0.60521739130434782, 0.92695652173913046, 0.9237681159420
2901, 0.9301449275362319, 0.93710144927536232, 0.92985507246376808, 0.93
275362318840582, 0.92985507246376808, 0.93391304347826087, 0.93652173913
043479]
    # print("Stochastic gradient descent on standardized data")
    # sgd_standardized_training_loss = training_loss(standardize(X_trai
n), y_train, train_sgd, alpha=1e-4)
    sgd_standardized = [0.65884057971014498, 0.9040579710144927, 0.90869
565217391302, 0.90927536231884054, 0.91130434782608694, 0.91130434782608
694, 0.9104347826086957, 0.91130434782608694, 0.91072463768115941, 0.911
30434782608694]
    # print("Stochastic gradient descent on binarized data")
    # sgd_binarized_training_loss = training_loss(binarize(X_train), y_t
rain, train_sgd, alpha=1e-3)
    sgd_binarized = [0.39478260869565218, 0.89681159420289858, 0.8886956
52173913, 0.89217391304347826, 0.8863768115942029, 0.88579710144927537,
0.89478260869565218, 0.89101449275362321, 0.89130434782608692, 0.8811594
2028985506]

    """ Training losses for stochastic gradient descenet with decaying a
lpha """
    # print("Decaying Alpha Stochastic gradient descent on log lifted da
ta")
    # sgda_log_training_loss = training_loss(log_lift(X_train), y_train,
 train_sgd_decay, alpha=1e-3, reg=0.01)
    # print("Training Losses: " + str(sgda_log_training_loss))
    sgda_log = [0.60521739130434782, 0.60521739130434782, 0.605217391304
34782, 0.60521739130434782, 0.60521739130434782, 0.60521739130434782, 0.
60521739130434782, 0.60521739130434782, 0.60521739130434782, 0.605217391
30434782]
    # print("Decaying Alpha Stochastic gradient descent on standardized
 data")
    # sgda_standardized_training_loss = training_loss(standardize(X_trai
n), y_train, train_sgd_decay, alpha=1e-3, reg=0.01)
    # print("Training Losses: " + str(sgda_standardized_training_loss))
```

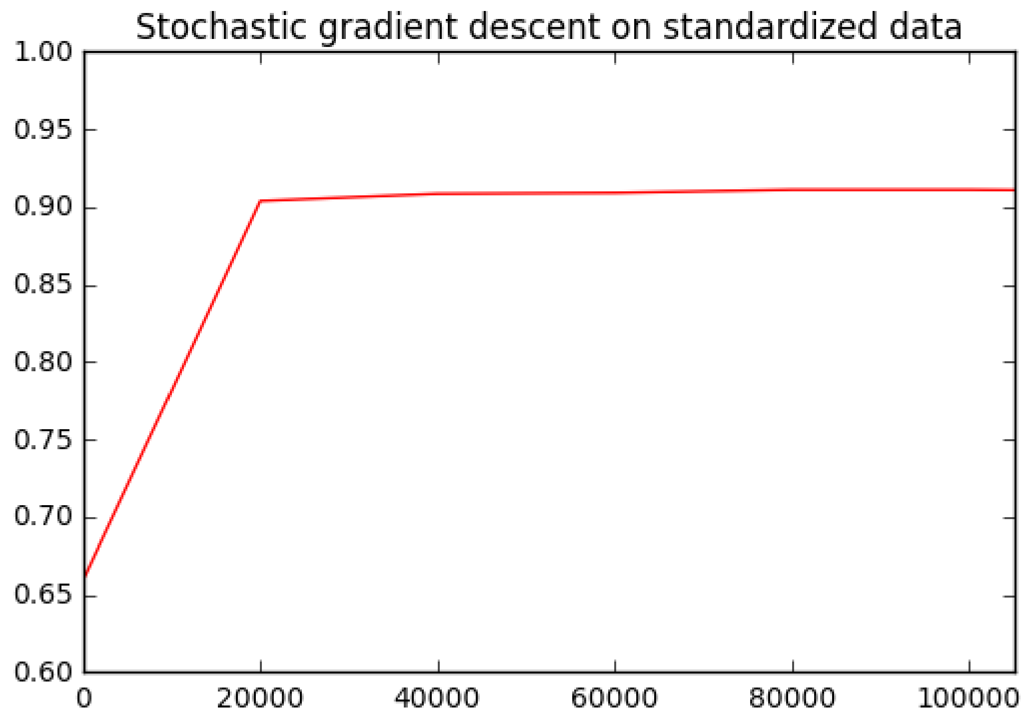
```
    sgda_standardized = [0.65884057971014498, 0.88144927536231887, 0.892
75362318840579, 0.87449275362318846, 0.88144927536231887, 0.894202898550
72466, 0.87855072463768114, 0.888695652173913, 0.88550724637681155, 0.89
507246376811589]
    # print("Decaying Alpha Stochastic gradient descent on binarized dat
a")
    # sgda_binarized_training_loss = training_loss(binarize(X_train), y_
train, train_sgd_decay, alpha=1e-3, reg=0.01)
    # print("Training Losses: " + str(sgda_binarized_training_loss))
    sgda_binarized = [0.39478260869565218, 0.70028985507246377, 0.864057
97101449278, 0.87333333333333329, 0.60521739130434782, 0.878550724637681
14, 0.83043478260869563, 0.71739130434782605, 0.82521739130434779, 0.718
5507246376811]

    """ Plotting """
    plot_training_loss(log, "Batch gradient descent on log lifted data")
    plot_training_loss(standardized, "Batch gradient descent on standard
ized data")
    plot_training_loss(binarized, "Batch gradient descent on binarized d
ata")
    plot_training_loss(sgd_log, "Stochastic gradient descent on log lift
ed data")
    plot_training_loss(sgd_standardized, "Stochastic gradient descent on
 standardized data")
    plot_training_loss(sgd_binarized, "Stochastic gradient descent on bi
narized data")
    plot_training_loss(sgda_log, "Stochastic gradient descent with decay
ing alpha on log lifted data")
    plot_training_loss(sgda_standardized, "Stochastic gradient descent w
ith decaying alpha on standardized data")
    plot_training_loss(sgda_binarized, "Stochastic gradient descent with
 decaying alpha on binarized data")

    """ Kaggle Test """
    model = train_gd(log_lift(X_train), y_train, alpha=1e-7, reg=0.1, nu
m_iter=100000)
    pred_y_test = predict(model, log_lift(X_test))

    c = csv.writer(open("hw3_kaggle.csv", "wt"))
    c.writerow(['Id', 'Category'])
    for i, val in enumerate(pred_y_test):
      c.writerow((i+1, int(val)))
```

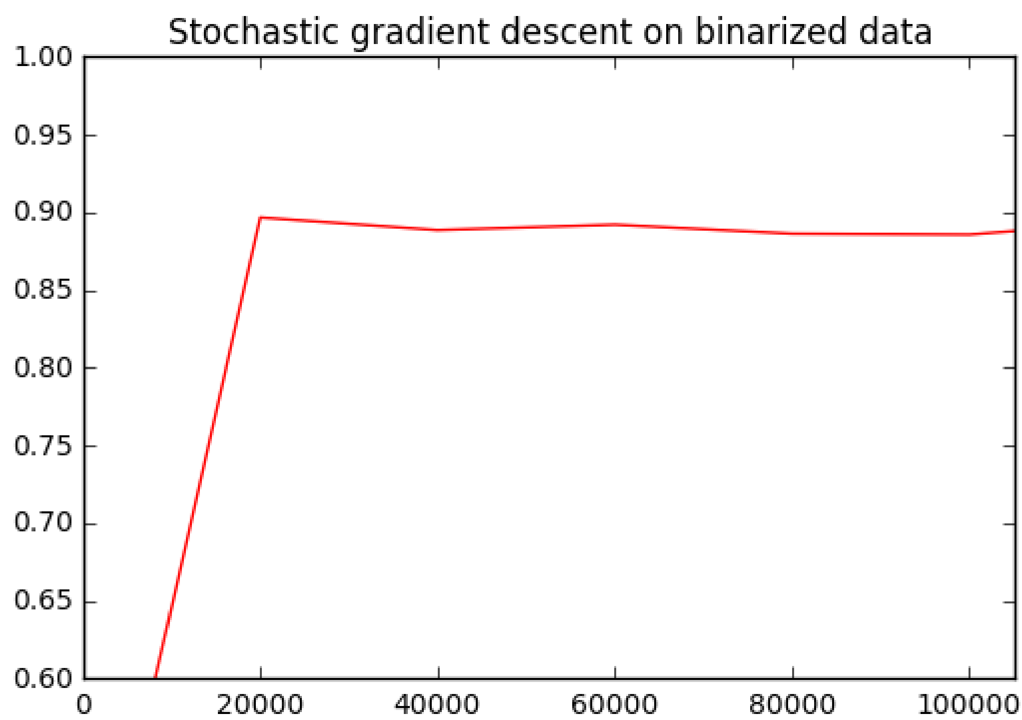

Batch gradient descent on log lifted data
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
20000
40000
60000
80000
100000

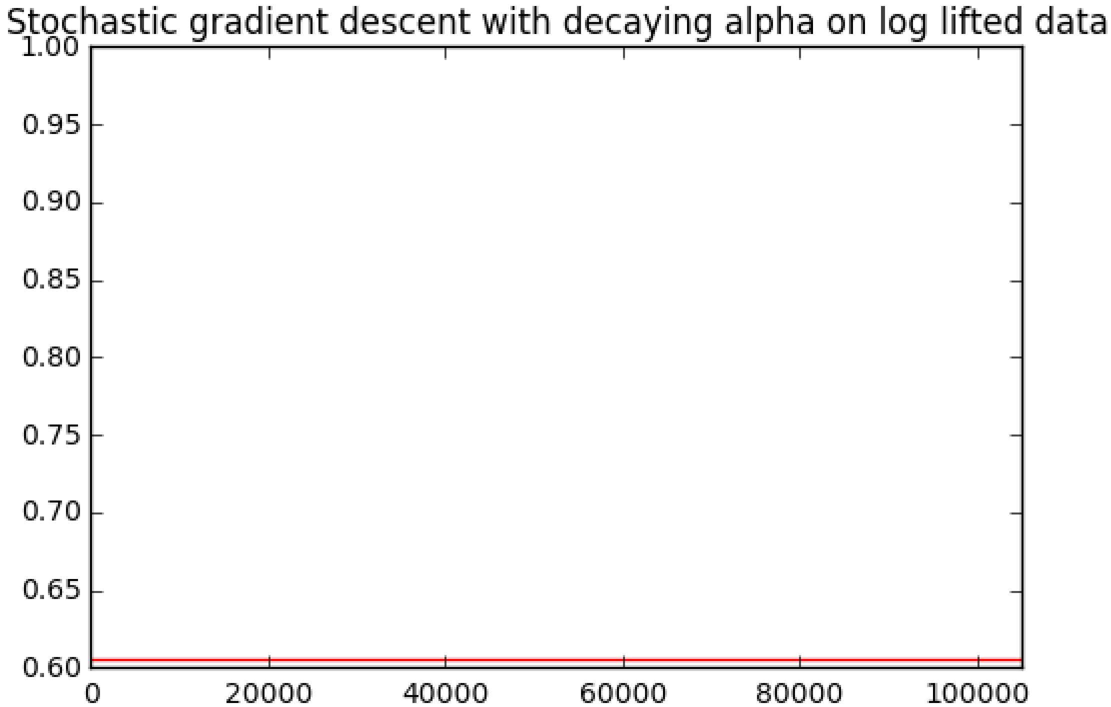

Batch gradient descent on standardized data
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
20000
40000
60000
80000
100000

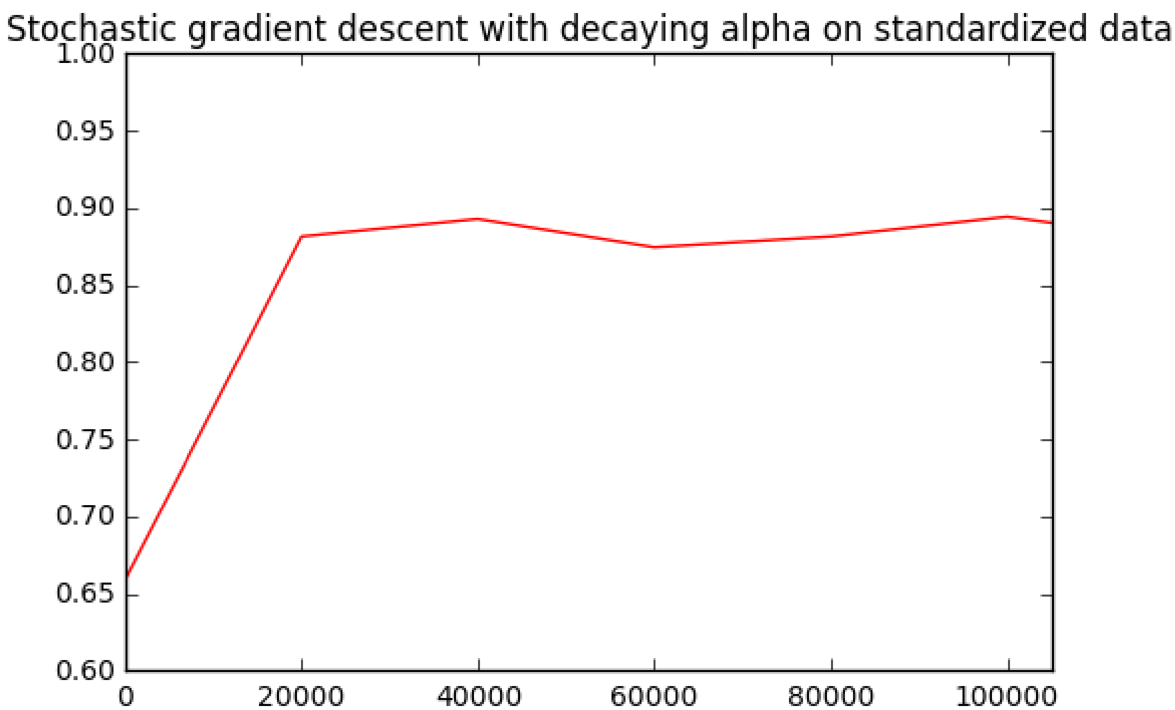

Batch gradient descent on binarized data
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
20000
40000
60000
80000
100000

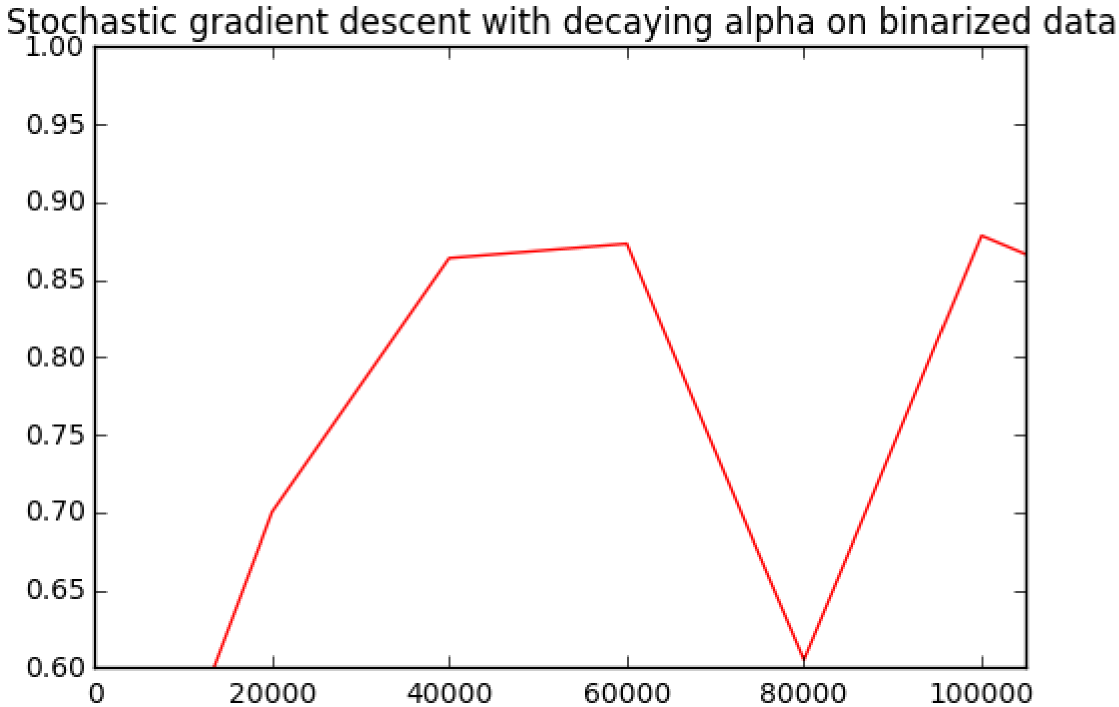

Stochastic gradient descent on log lifted data
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
20000
40000
60000
80000
100000


Stochastic gradient descent on standardized data
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
20000
40000
60000
80000
100000


Stochastic gradient descent on binarized data
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
20000
40000
60000
80000
100000


Stochastic gradient descent with decaying alpha on log lifted data
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
20000
40000
60000
80000
100000


Stochastic gradient descent with decaying alpha on standardized data
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
20000
40000
60000
80000
100000


Stochastic gradient descent with decaying alpha on binarized data
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
20000
40000
60000
80000
100000

```
In [ ]:
```