Bayesian Models for Machine Learning
Problem Set #3

Si Kai Lee sl3950@columbia.edu

November 10, 2016

Problem 1

We want to find the distribution $q(w, \alpha_{1:d}, \lambda) = q(w)q(\lambda)\prod_{k=1}^{d} q(\alpha_k)$ that best approximates $p(w, \alpha_{1:d}, \lambda|Y, X)$. Assume that $p(w, \alpha_{1:d}, \lambda|Y, X) = p(Y|w, \alpha_{1:n}, \lambda, X)p(w|\alpha_{1:n})p(\lambda)\prod_{k=1}^{d} p(\alpha_k)$.

a

We find the optimal distribution for each q distribution.

$$
\begin{aligned}
q(\lambda) &\propto \exp\left\{\mathbb{E}_{-q(\lambda)}[\ln p(Y|w, \alpha_{1:n}, \lambda, X)] + \ln q(\lambda)\right\} \\
&\propto \exp\left\{\mathbb{E}_{q(w)}[\frac{n}{2}\ln\lambda - \frac{\lambda}{2}(Y - X^T w)^T(Y - X^T w)] + (e_0 - 1)\ln\lambda - f_0\lambda\right\} \\
&\propto \exp\left\{(\frac{n}{2} + e_0 - 1)\ln\lambda - \frac{1}{2}(Y^TY - 2Y^TX^T\mathbb{E}_{q(w)}[w] + tr(\mathbb{E}_{q(w)}[ww^T]XX^T) + 2f_0)\lambda\right\} \\
&= Gamma(e_0', f_0') \text{ where } e_0' = \frac{n}{2} + e_0,\ f_0' = \frac{1}{2}(Y^TY - 2Y^TX^T\mathbb{E}_{q(w)}[w] + tr(\mathbb{E}_{q(w)}[ww^T]XX^T) + 2f_0) \\
q(\alpha_k) &\propto \exp\left\{\mathbb{E}_{-q(\alpha_k)}[\ln p(w_k|\alpha_k)] + \ln q(\alpha_k)\right\} \\
&\propto \exp\left\{\mathbb{E}_{q(w_k)}[\frac{1}{2}\ln\alpha_k - \frac{\alpha_k}{2}(w_k^2)] + (a_0 - 1)\ln\alpha_k - b_0\alpha_k\right\} \\
&\propto \exp\left\{(\frac{1}{2} + a_0 - 1)\ln\alpha_k - \frac{1}{2}(\mathbb{E}_{q(w_k)}[w_k^2] + 2b_0)\alpha_k\right\} \\
&= Gamma(a_0^{(k)'}, b_0^{(k)'}) \text{ where } a_0^{(k)'} = \frac{1}{2} + a_0,\ b_0^{(k)'} = \frac{1}{2}(\mathbb{E}_{q(w_k)}[w_k^2] + 2b_0) \\
q(w) &\propto \exp\left\{\mathbb{E}_{-q(w)}[\ln p(Y|w, \alpha_{1:n}, \lambda, X) + \ln q(w)]\right\} \\
&\propto \exp\left\{\mathbb{E}_{q(\lambda)}[-\frac{\lambda}{2}(Y - X^Tw)^T(Y - X^Tw)] + \mathbb{E}_{q(\alpha_{1:d})}[-\frac{A}{2}W^TW)\right\} \text{ where } A = diag(\alpha_1, ..., \alpha_d) \\
&\propto \exp\left\{-\frac{1}{2}w^T\underbrace{(\mathbb{E}_{q(\lambda)}[\lambda]XX^T + \mathbb{E}_{q(\alpha_{1:d})}[A])}_{M}w - 2w^T\mathbb{E}_{q(\lambda)}[\lambda]XY\right\} \\
&\propto \exp\left\{-\frac{1}{2}(w^TMw - 2w^TM\mathbb{E}_{q(\lambda)}[\lambda]M^{-1}XY + (\mathbb{E}_{q(\lambda)}[\lambda]M^{-1}XY)^TM(\mathbb{E}_{q(\lambda)}[\lambda]M^{-1}XY))\right\} \\
&= \mathcal{N}(\mathbb{E}_{q(\lambda)}[\lambda]M^{-1}XY, M^{-1})
\end{aligned}
$$

The following expectations are defined to allow the q distributions to be updated:

- $\mathbb{E}_{q(w)}[w] = \mathbb{E}_{q(\lambda)}[\lambda]M^{-1}XY$
- $\mathbb{E}_{q(w_k)}[w_k^2] = (\mathbb{E}_{q(w)}[w])_k^2 + M_{kk}^{-1}$
- $\mathbb{E}_{q(w)}[ww^T] = M^{-1} + \mathbb{E}_{q(w)}[w]\mathbb{E}_{q(w)}[w]^T$
- $\mathbb{E}_{q(\lambda)}[\lambda] = e_0'/f_0'$
- $\mathbb{E}_{q(\alpha_{1:d})}[A] = diag(a_0^{(1)'}/b_0^{(1)'}, ..., a_0^{(d)'}/b_0^{(d)'})$

b

A VI algorithm for Bayesian Regression

1. Initialise $a_0', b_0', e_0', f_0', \mu_0'$ and Σ_0'.
2. For iteration $t = 1, ..., T$:
   (a) Update $q(\lambda)$ by setting
   - $e_t' = \frac{n}{2} + e_0$
   - $f_t' = \frac{1}{2}(Y^TY - 2Y^TX^T\mathbb{E}_{q(w)}[w_t] + tr(\mathbb{E}_{q(w)}[w_tw_t^T]XX^T) + 2f_0)$

   where $\mathbb{E}_{q(w)}[w_t] = \mu_{t-1}$ and $\mathbb{E}_{q(w)}[w_tw_t^T] = \Sigma_{t-1} + \mathbb{E}_{q(w)}[w_t]\mathbb{E}_{q(w)}[w_t]^T$

   (b) Update each $q(\alpha_k)$ by setting
   - $a_t^{(k)'} = \frac{1}{2} + a_0$
   - $b_t^{(k)'} = \frac{1}{2}(\mathbb{E}_{q(w_k)}[w_{kt}^2] + 2b_0)$

   where $\mathbb{E}_{q(w_k)}[w_{kt}^2] = (\mathbb{E}_{q(w)}[w_t])_k^2 + (\Sigma_{t-1})_{kk}$

   (c) Update $q(w)$ by setting
   - $M_t' = (\mathbb{E}_{q(\lambda)}[\lambda_t]XX^T + \mathbb{E}_{q(\alpha_{1:d})}[A_t])$
   - $\Sigma_t' = M_t^{-1}$
   - $\mu_t' = \mathbb{E}_{q(\lambda)}[\lambda_t]M_t^{-1}XY$

   where $\mathbb{E}_{q(\lambda)}[\lambda_t] = e_t'/f_t'$ and $\mathbb{E}_{q(\alpha_{1:d})}[A_t] = diag(a_t^{(1)'}/b_t^{(1)'}, ..., a_t^{(d)'}/b_t^{(d)'})$

   (d) Assess convergence by calculating $\mathcal{L}(a_t', b_t', e_t', f_t', \mu_t', \Sigma_t')$

c

$$\mathcal{L}((a_t^{(1)'}, b_t^{(1)'}), ..., (a_t^{(d)'}, a_t^{(d)'}), e_t', f_t', \mu_t', \Sigma_t')$$
$$= \int_w \int_\lambda \int_{\alpha_{1:d}} q(w)q(\lambda)\prod_{k=1}^d q(\alpha_k) \ln \frac{p(w, \alpha_{1:d}, \lambda|Y, X)}{q(w)q(\lambda)\prod_{k=1}^d q(\alpha_k)} d\alpha_{1:d}d\lambda dw$$
$$= \int_w \int_{\alpha_{1:d}} q(w)\ln p(w)d\alpha_{1:d}dw + \int_\lambda q(\lambda)\ln p(\lambda)d\lambda + \int_{\alpha_{1:d}} \prod_{k=1}^d q(\alpha_k)\ln p(\alpha_k)d\alpha_{1:d} +$$
$$\int_w \int_\lambda q(w)q(\lambda)\ln p(Y|w, \lambda, X)d\lambda dw - \int_w q(w)\ln q(w)dw - \int_\lambda q(\lambda)\ln q(\lambda)d\lambda - \int_{\alpha_{1:d}} \prod_{k=1}^d q(\alpha_k)\ln q(\alpha_k)d\alpha_{1:d}$$

As the above equation is complicated, we look at each term individually.

$$
\begin{aligned}
\int_w \int_{\alpha_{1:d}} q(w) \ln p(w) dw &= -\frac{d}{2}\ln 2\pi + \frac{1}{2}\sum_{i=k}^{d} \mathbb{E}_{\alpha_{1:d}}[\ln \alpha_k] - \frac{1}{2}\mathbb{E}_{q(w),q(\alpha_{1:d})}[w_t^T A w_t] \\
&= -\frac{d}{2}\ln 2\pi + \frac{1}{2}\sum_{i=k}^{d}(\psi(a_t^{(k)'}) - \ln b_t^{(k)'}) - \frac{1}{2}tr((\mu_t' {\mu_t'}^T + \Sigma_t')\mathbb{E}_{q(\alpha_{1:d})}A) \\
\int_\lambda q(\lambda)\ln p(\lambda) d\lambda &= e_0 \ln f_0 - \ln\Gamma(e_0) + (e_0 - 1)\mathbb{E}_{q(\lambda)}[\ln\lambda] - f_0\mathbb{E}_{q(\lambda)}[\lambda] \\
&= e_0 \ln f_0 - \ln\Gamma(e_0) + (e_0 - 1)(\psi(e_t') - \ln f_t') - f_0\frac{e_t'}{f_t'} \\
\int_{\alpha_{1:d}} \prod_{k=1}^{d} q(\alpha_k)\ln p(\alpha_k) d\alpha_{1:d} &= \sum_{k=1}^{d}\int_{\alpha_k}(\alpha_k)\ln p(\alpha_k) d\alpha_k \\
&= \sum_{k=1}^{d} a_0 \ln b_0 - \ln\Gamma(a_0) + (a_0 - 1)(\psi(a_t^{(k)'}) - \ln b_t^{(k)'}) - b_0\frac{a_t^{(k)'}}{b_t^{(k)'}} \\
\int_w q(w)\ln q(w) dw &= \frac{1}{2}\ln((2\pi e)^n|\Sigma_t|) \\
\int_\lambda q(\lambda)\ln q(\lambda) d\lambda &= e_t' \ln f_t' - \ln\Gamma(e_t') + (e_t' - 1)(\psi(e_t') - \ln f_t') - f_t'\frac{e_t'}{f_t'} \\
&= \ln f_t' - \ln\Gamma(e_t') + (e_t' - 1)\psi(e_t') - e_t' \\
\int_{\alpha_{1:d}} \prod_{k=1}^{d} q(\alpha_k)\ln q(\alpha_k) d\alpha_{1:d} &= \sum_{k=1}^{d} \ln b_t^{(k)'} - \ln\Gamma(a_t^{(k)'}) + (a_t^{(k)'} - 1)\psi(a_t^{(k)'}) - a_t^{(k)'}
\end{aligned}
$$

And lastly,

$$
\begin{aligned}
&\int_w \int_\lambda q(w)q(\lambda)\ln p(Y|w,\lambda,X) d\lambda dw \\
&= -\frac{n}{2}\ln 2\pi + \frac{n}{2}\mathbb{E}_{q(\lambda)}[\ln\lambda] - \frac{\mathbb{E}_{q(\lambda)}[\lambda]}{2}\mathbb{E}_{q(w)}[(Y - X^T w)^T(Y - X^T w)] \\
&= -\frac{n}{2}\ln 2\pi + \frac{n}{2}(\psi(e_t') - \ln f_t') - \frac{e_t'}{2f_t'}(Y^T Y - 2Y^T X^T \mu_t' - tr((\Sigma_t' + \mu_t' {\mu_t'}^T)XX^T))
\end{aligned}
$$

In [1]:

```
# Use gammaln for stability
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
from scipy.io import loadmat
from scipy.special import digamma, gammaln
```

In [2]:

```
data_1 = loadmat('hw3_data_mat/data1.mat')
data_2 = loadmat('hw3_data_mat/data2.mat')
data_3 = loadmat('hw3_data_mat/data3.mat')
```

In [3]:

```
# Set prior parameters
a_0 = b_0 = 10e-16
e_0 = f_0 = 1
```

In [4]:

```
# Update q(lambda)
def update_q_lambda(n, e_0, f_0, mu, sigma, Y_T_Y, Y_T_X_T, X_X_T):
    e_p = 0.5 * n + e_0
    E_w_w_T = sigma + np.dot(mu, mu.T)
    f_p = 0.5 * (Y_T_Y - 2 * np.dot(Y_T_X_T, mu) + np.trace(np.dot(E_w_w
_T, X_X_T))) + f_0
    return e_p, float(f_p)
```

In [5]:

```
# Update q(alpha_k)
def update_q_alpha_k(a_0, b_0, mu, sigma, k):
    a_kp = 0.5 + a_0
    E_w_k_2 = mu[k]**2 + sigma[k, k]
    b_kp = 0.5 * E_w_k_2 + b_0
    return a_kp, b_kp
```

In [6]:

```
# Update q(w)
def update_q_w(e_t, f_t, a_t, b_t, X_X_T, X, Y):
    A_p = a_t / b_t
    E_lambda = e_t/float(f_t)
    M_p = E_lambda * X_X_T + np.diag(A_p)
    sigma_p = np.linalg.inv(M_p)
    mu_p = E_lambda * np.dot(np.dot(sigma_p, X.T), Y)
    return mu_p, sigma_p
```

In [7]:

```
# Compute L_1
def L_1(dim, a_t, b_t, mu_t, sigma_t):
    A_t = a_t / b_t
    E_A = np.diag(A_t)
    E_w_T_w = sigma_t + np.dot(mu_t, mu_t.T)
    E_ln_alpha = map(lambda a_b: digamma(a_b[0]) - np.log(a_b[1]), zip(a
_t, b_t))
    return - 0.5 * dim * np.log(2 * np.pi) + 0.5 * sum(E_ln_alpha) - 0.5
 * np.trace(np.dot(E_w_T_w, E_A))
```

In [8]:
```
# Compute L_2
def L_2(e_0, f_0, e_t, f_t):
    return e_0 * np.log(f_0) - gammaln(e_0) + (e_0 -1) * (digamma(e_t) -
 np.log(f_t)) - f_0 * e_t/float(f_t)
```

In [9]:
```
# Compute L_3
def L_3(a_0, b_0, a_t, b_t):
    return sum(map(lambda a_b: a_0 * np.log(b_0) - gammaln(a_0) + (a_0 -
 1) * (digamma(a_b[0]) - np.log(a_b[1])) \
                  - b_0 * a_b[0]/float(a_b[1]), zip(a_t, b_t)))
```

In [10]:
```
# Compute L_4
def L_4(dim, sigma_t):
    # Compute in logspace to prevent overflow
    sign, logdet_sigma_t = np.linalg.slogdet(sigma_t)
    # As 2 * np.pi * np.exp(1))** dim) is a constant and causes overflo
w, remove
    return - 0.5 * (sign * logdet_sigma_t)
```

In [11]:
```
# Compute L_5
def L_5(e_t, f_t):
    return np.log(f_t) - gammaln(e_t) + (e_t - 1) * digamma(e_t) - e_t
```

In [12]:
```
# Compute L_6
def L_6(a_t, b_t):
    return sum(map(lambda a_b: np.log(a_b[1]) - gammaln(a_b[0]) +
(a_b[0] - 1) * digamma(a_b[0]) - a_b[0], zip(a_t, b_t)))
```

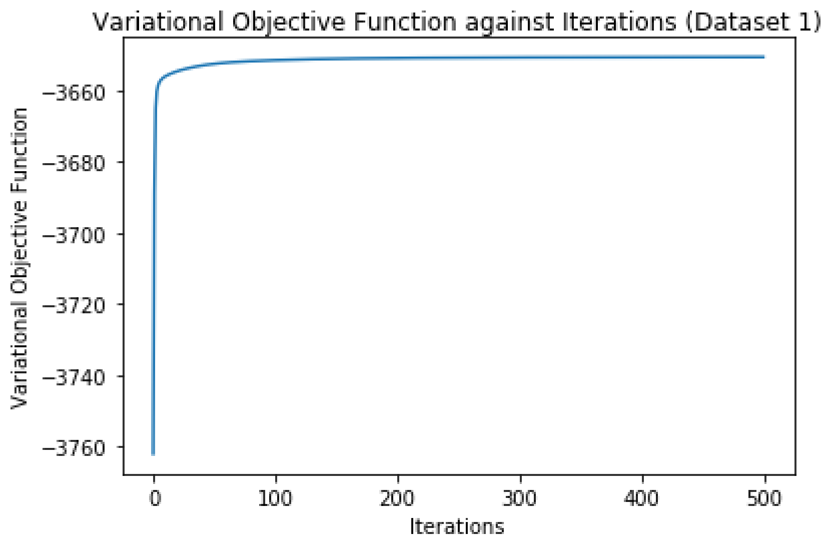
In [13]:
```
# Compute L_7
def L_7(n, e_t, f_t, mu_t, sigma_t, Y_T_Y, Y_T_X_T, X_X_T):
    E_ln_lambda = digamma(e_t) - np.log(f_t)
    E_lambda = e_t / float(f_t)
    E_w_T_w = np.dot(mu_t, mu_t.T) + sigma_t
    return 0.5 * n * (E_ln_lambda - np.log(2 * np.pi)) \
           - 0.5 * E_lambda * (Y_T_Y - 2 * np.dot(Y_T_X_T, mu_t) + np.tr
ace(np.dot(E_w_T_w, X_X_T)))
```

In [14]:
```
def variational_inference(X, y):
    # Get dimensions of X
    dim = X.shape[1]
    n = X.shape[0]

    # Calculate variables
    Y_T_Y = np.dot(y.T, y)[0][0]
    Y_T_X_T = np.dot(y.T, X)
    X_X_T = np.dot(X.T, X)

    # Initialise variables
    a_t = np.array([a_0] * dim, dtype='float64')
    b_t = np.array([b_0] * dim, dtype='float64')
    e_t = e_0
    f_t = f_0
    mu_t = np.zeros(dim, dtype='float64')
    sigma_t = np.diag(np.ones(dim, dtype='float64'))
    L = []
    L_1_t = L_2_t = L_3_t = L_4_t = L_5_t = L_6_t = L_7_t = 0

    # Run VI algorithm
    for t in range(500):
        # print t
        e_t, f_t = update_q_lambda(n, e_0, f_0, mu_t, sigma_t, Y_T_Y, Y_
T_X_T, X_X_T)
        a_p = []
        b_p = []
        for k in range(dim):
            a_kp, b_kp = update_q_alpha_k(a_0, b_0, mu_t, sigma_t, k)
            a_p.append(a_kp)
            b_p.append(b_kp)
        a_t = np.array(a_p)
        b_t = np.array(b_p).flatten()
        mu_t, sigma_t = update_q_w(e_t, f_t, a_t, b_t, X_X_T, X, y)
        L_1_t = L_1(dim, a_t, b_t, mu_t, sigma_t)
        L_2_t = L_2(e_0, f_0, e_t, f_t)
        L_3_t = L_3(a_0, b_0, a_t, b_t)
        L_4_t = L_4(dim, sigma_t)
        L_5_t = L_5(e_t, f_t)
        L_6_t = L_6(a_t, b_t)
        L_7_t = L_7(n, e_t, f_t, mu_t, sigma_t, Y_T_Y, Y_T_X_T, X_X_T)
        # print L_1_t, L_2_t, L_3_t, L_4_t, L_5_t, L_6_t, L_7_t
        L.append((L_1_t + L_2_t + L_3_t + L_7_t - L_4_t - L_5_t - L_6_t)
[0][0])

    return L, a_t, b_t, e_t, f_t, mu_t, sigma_t, dim
```

In [15]:
```
LL_1, a_1, b_1, e_1, f_1, mu_1, sigma_1, dim_1 = variational_inference(d
ata_1['X'], data_1['y'])
```

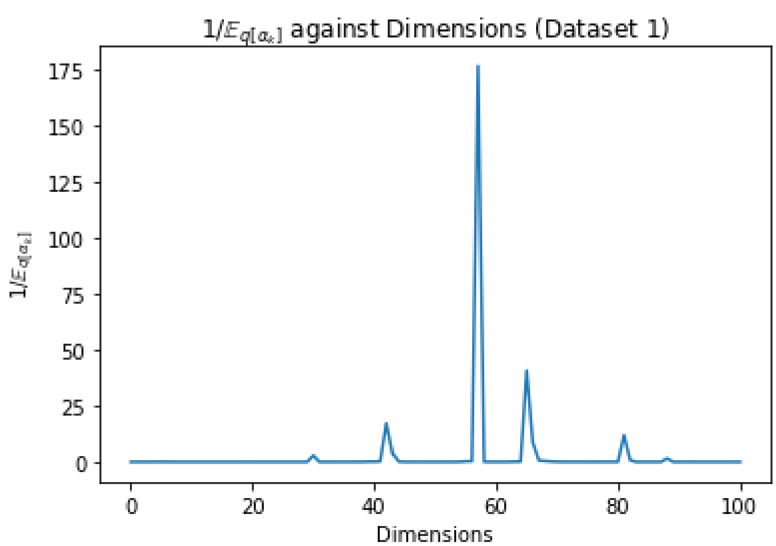
In [16]:
```
plt.plot(range(500), LL_1)
plt.xlabel('Iterations')
plt.ylabel('Variational Objective Function')
plt.title('Variational Objective Function against Iterations (Dataset
 1)')
```

Out[16]:
```
Text(0.5,1,u'Variational Objective Function against Iterations (Dataset
 1)')
```



In [17]:
```
plt.plot(range(dim_1), 1/(a_1/b_1))
plt.xlabel('Dimensions')
plt.ylabel(r'$1/\mathbb{E}_{q[\alpha_k]}$')
plt.title(r'$1/\mathbb{E}_{q[\alpha_k]}$ against Dimensions (Dataset
1)')
```

Out[17]:
```
Text(0.5,1,u'$1/\\mathbb{E}_{q[\\alpha_k]}$ against Dimensions (Dataset
 1)')
```

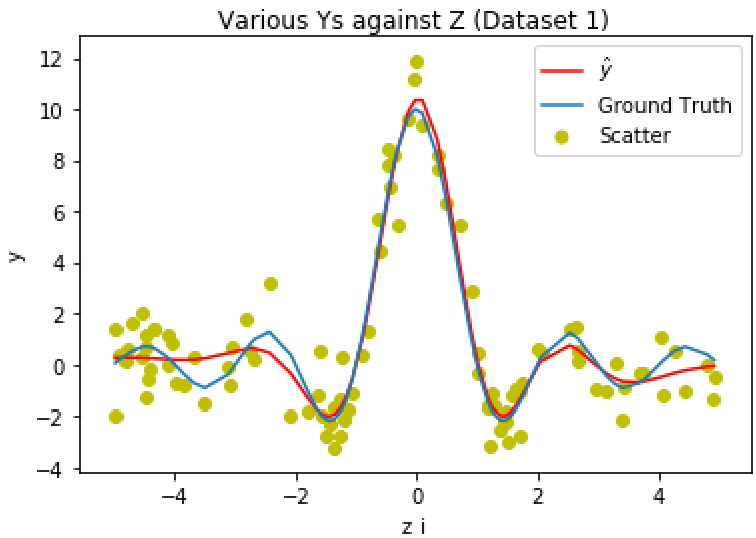


In [18]:
```
print '1/E_q[lambda] (Dataset 1) =', f_1/e_1
```

```
1/E_q[lambda] (Dataset 1) = 1.08004906537
```

In [19]:
```
y_hat = np.dot(data_1['X'], mu_1)
plt.plot(data_1['z'], y_hat, 'r', label=r'$\hat{y}$')
plt.scatter(data_1['z'], data_1['y'], c='y', label='Scatter')
plt.plot(data_1['z'], 10 * np.sinc(data_1['z']), label='Ground Truth')
plt.legend()
plt.xlabel('z_i')
plt.ylabel('y')
plt.title('Various Ys against Z (Dataset 1)')
```

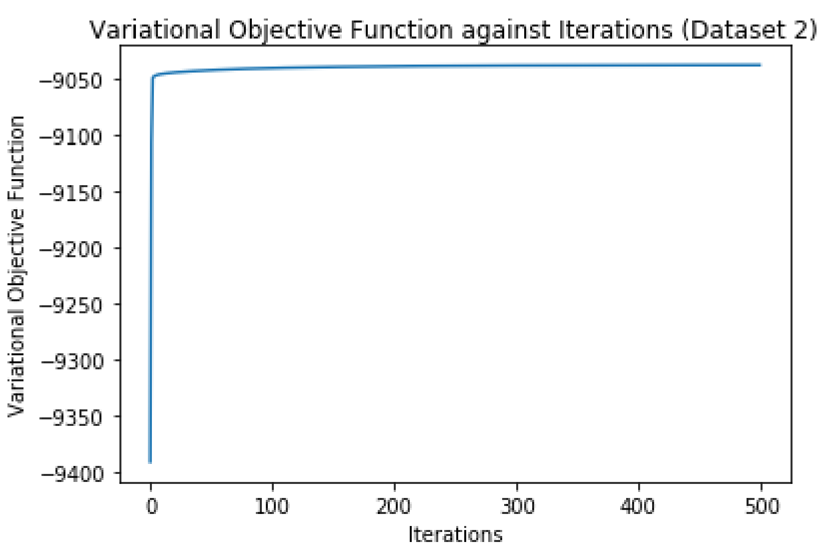
Out[19]:
```
Text(0.5,1,u'Various Ys against Z (Dataset 1)')
```



In [20]:
```
LL_2, a_2, b_2, e_2, f_2, mu_2, sigma_2, dim_2 = variational_inference(d
ata_2['X'].astype('float64'), data_2['y'].astype('float64'))
```

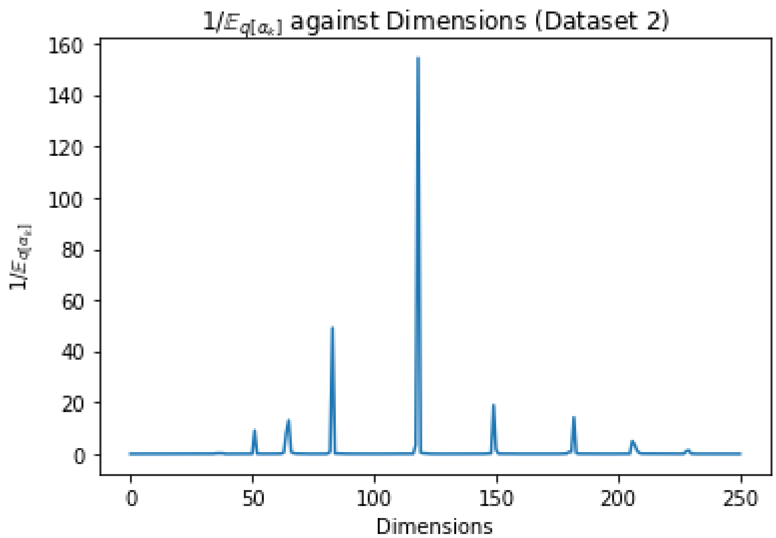
In [21]:
```
plt.plot(range(500), LL_2)
plt.xlabel('Iterations')
plt.ylabel('Variational Objective Function')
plt.title('Variational Objective Function against Iterations (Dataset
 2)')
```

Out[21]:
```
Text(0.5,1,u'Variational Objective Function against Iterations (Dataset
 2)')
```



In [22]:
```
plt.plot(range(dim_2), 1/(a_2/b_2))
plt.xlabel('Dimensions')
plt.ylabel(r'$1/\mathbb{E}_{q[\alpha_k]}$')
plt.title(r'$1/\mathbb{E}_{q[\alpha_k]}$ against Dimensions (Dataset
2)')
```

Out[22]:
```
Text(0.5,1,u'$1/\\mathbb{E}_{q[\\alpha_k]}$ against Dimensions (Dataset
 2)')
```

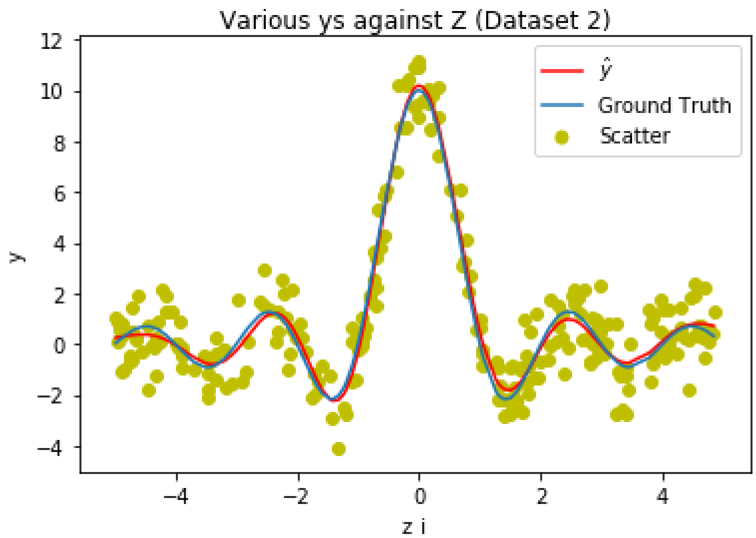


In [23]:
```
print '1/E_q[lambda] (Dataset 2) =', f_2/e_2
```

```
1/E_q[lambda] (Dataset 2) = 0.89944378672
```

In [24]:
```
y_hat = np.dot(data_2['X'], mu_2)
plt.plot(data_2['z'], y_hat, 'r', label=r'$\hat{y}$')
plt.scatter(data_2['z'], data_2['y'], c='y', label='Scatter')
plt.plot(data_2['z'], 10 * np.sinc(data_2['z']), label='Ground Truth')
plt.legend()
plt.xlabel('z_i')
plt.ylabel('y')
plt.title('Various ys against Z (Dataset 2)')
```

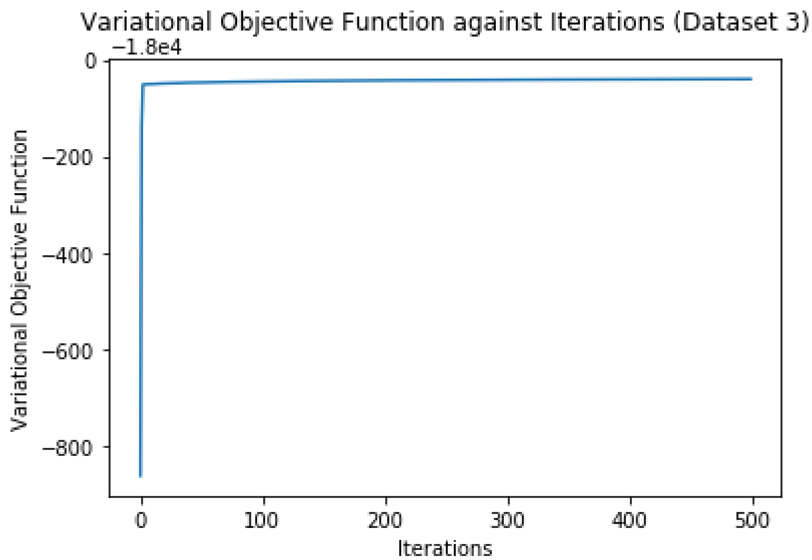
Out[24]:
```
Text(0.5,1,u'Various ys against Z (Dataset 2)')
```



In [25]:
```
LL_3, a_3, b_3, e_3, f_3, mu_3, sigma_3, dim_3 = variational_inference(d
ata_3['X'].astype('float64'), data_3['y'].astype('float64'))
```

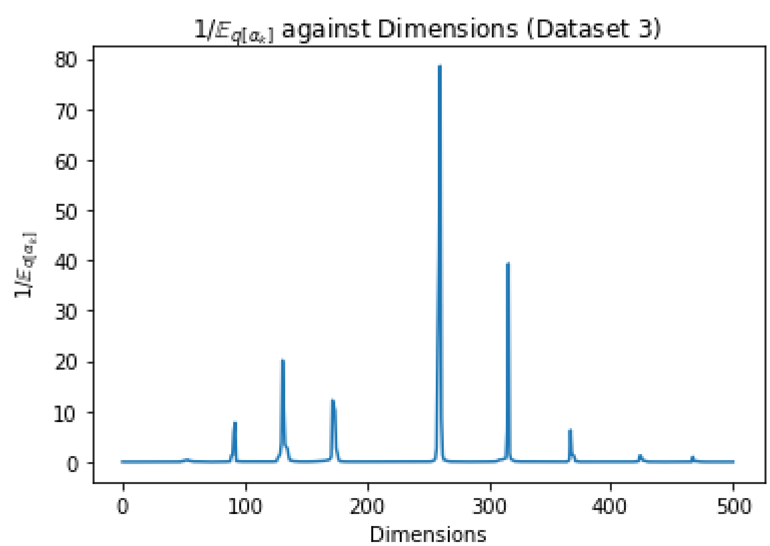
In [26]:
```
plt.plot(range(500), LL_3)
plt.xlabel('Iterations')
plt.ylabel('Variational Objective Function')
plt.title('Variational Objective Function against Iterations (Dataset
 3)', y=1.05)
```

Out[26]:
```
Text(0.5,1.05,u'Variational Objective Function against Iterations (Data
set 3)')
```



In [27]:
```
plt.plot(range(dim_3), 1/(a_3/b_3))
plt.xlabel('Dimensions')
plt.ylabel(r'$1/\mathbb{E}_{q[\alpha_k]}$')
plt.title(r'$1/\mathbb{E}_{q[\alpha_k]}$ against Dimensions (Dataset
3)')
```

Out[27]:
```
Text(0.5,1,u'$1/\\mathbb{E}_{q[\\alpha_k]}$ against Dimensions (Dataset
 3)')
```



In [28]:
```
print '1/E_q[lambda] (Dataset 3) =', f_3/e_3
```

```
1/E_q[lambda] (Dataset 3) = 0.978150754147
```

In [29]:
```
y_hat = np.dot(data_3['X'], mu_3)
plt.plot(data_3['z'], y_hat, 'r', label=r'$\hat{y}$')
plt.scatter(data_3['z'], data_3['y'], c='y', label='Scatter')
plt.plot(data_3['z'], 10 * np.sinc(data_3['z']), label='Ground Truth')
plt.legend()
plt.xlabel('z_i')
plt.ylabel('y')
plt.title('Various Ys against Z (Dataset 3)')
```

Out[29]:
```
Text(0.5,1,u'Various Ys against Z (Dataset 3)')
```