Bayesian Models for Machine Learning
Problem Set #2

Si Kai Lee sl3950@columbia.edu

October 21, 2016

Problem 1

We want to find $W' = \underset{W}{\arg\max} p(x_{1:n}, W)$. Split $\ln p(x_{1:n}, W) = \sum_{i=1}^{n} \ln p(x_i, W)$ and expand $\ln p(x_i, W)$ to obtain $\ln p(x_i, W) = \ln p(x_i, W, z_i) - \ln p(z_i|x_i, W)$. We also assume that the joint $p(x_i, W, z_i)$ can be represented as $p(x_i, W, z_i) = p(w)p(x_i|z_i, W)p(z_i)$

The Expectation-Maximisation (EM) equation is derived as:

$$\begin{aligned}
\ln p(x_{1:n}, W) &= \sum_{i=1}^{n} \ln p(x_i, W, z_i) - \sum_{i=1}^{n} \ln p(z_i|x_i, W) \\
&= \sum_{i=1}^{n} \int_{q(\phi)} q(\phi) \frac{\ln p(x_i, W, z_i)}{q(\phi)} d\phi + \sum_{i=1}^{n} \int_{q(\phi)} q(\phi) \frac{q(\phi)}{\ln p(z_i|x_i, W)} d\phi \\
&= \mathbb{E}_{q(\phi)}[\ln p(x_i, W, z_i)] + KL(q||p)
\end{aligned}$$

We can express $\ln p(x_i, W, z_i)$ as the following:

$$\ln p(x_i, W, z_i) = \frac{dk}{2} \ln \frac{\lambda}{2\pi} - \frac{\lambda}{2} tr(W^T W) + \frac{d}{2} \ln \frac{1}{2\pi\sigma^2} - \frac{1}{2\sigma^2}(x_i - W z_i)^T (x_i - W z_i) + \frac{k}{2} \ln \frac{1}{2\pi} - \frac{1}{2} z_i^T z_i$$

Taking expectations w.r.t to $q(\phi)$, we have:

$$\begin{aligned}
\mathbb{E}_{q(\phi)}[\ln p(x_i, W, z_i)] &= \frac{dk}{2} \ln \frac{\lambda}{2\pi} - \frac{\lambda}{2} tr(W^T W) + \frac{d}{2} \ln \frac{1}{2\pi\sigma^2} - \frac{1}{2\sigma^2} \mathbb{E}_{q(\phi)}[x_i^T x_i - 2x_i^T W z_i + z_i^T W^T W z_i] \\
&+ \frac{k}{2} \ln \frac{1}{2\pi} - \frac{1}{2} \mathbb{E}_{q(\phi)}[z_i^T z_i] \\
&= \frac{dk}{2} \ln \frac{\lambda}{2\pi} - \frac{\lambda}{2} tr(W^T W) + \frac{d}{2} \ln \frac{1}{2\pi\sigma^2} - \frac{1}{2\sigma^2} x_i^T x_i + \frac{1}{\sigma^2} x_i^T W \mathbb{E}_{q(\phi)}[z_i] \\
&- \frac{1}{2\sigma^2} \mathbb{E}_{q(\phi)}[tr(z_i^T W^T W z_i)] - \frac{1}{2} \mathbb{E}_{q(\phi)}[tr(z_i^T z_i)] \\
&= \frac{dk}{2} \ln \frac{\lambda}{2\pi} - \frac{\lambda}{2} tr(W^T W) + \frac{d}{2} \ln \frac{1}{2\pi\sigma^2} - \frac{1}{2\sigma^2} x_i^T x_i + \frac{1}{\sigma^2} x_i^T W \mathbb{E}_{q(\phi)}[z_i] \\
&- \frac{1}{2\sigma^2} \mathbb{E}_{q(\phi)}[tr(z_i z_i^T W^T W)] - \frac{1}{2} \mathbb{E}_{q(\phi)}[tr(z_i z_i^T)] \\
&= \frac{dk}{2} \ln \frac{\lambda}{2\pi} - \frac{\lambda}{2} tr(W^T W) + \frac{d}{2} \ln \frac{1}{2\pi\sigma^2} - \frac{1}{2\sigma^2} x_i^T x_i + \frac{1}{\sigma^2} x_i^T W \mathbb{E}_{q(\phi)}[z_i] \\
&- \frac{1}{2\sigma^2} tr(\mathbb{E}_{q(\phi)}[z_i z_i^T] W^T W) - \frac{1}{2} tr(\mathbb{E}_{q(\phi)}[z_i z_i^T])
\end{aligned}$$

Maximise $\mathbb{E}_{q(\phi)} \ln p(x_i, W, z_i)$ w.r.t. to W

$$\begin{aligned}
\nabla_W \mathbb{E}_{q(\phi)} \ln p(x_i, W, z_i) &= -\lambda W + \frac{1}{\sigma^2} x_i^T \mathbb{E}_{q(\phi)}[z_i] - \frac{1}{\sigma^2} W \mathbb{E}_{q(\phi)}[z_i z_i^T] \\
0 &= -\lambda W + \frac{1}{\sigma^2} x_i^T \mathbb{E}_{q(\phi)}[z_i] - \frac{1}{\sigma^2} W \mathbb{E}_{q(\phi)}[z_i z_i^T] \\
x_i^T \mathbb{E}_{q(\phi)}[z_i] &= \sigma^2 \lambda W + W \mathbb{E}_{q(\phi)}[z_i z_i^T] \\
x_i^T \mathbb{E}_{q(\phi)}[z_i] &= W(\sigma^2 \lambda + \mathbb{E}_{q(\phi)}[z_i z_i^T])) \\
W &= (x_i^T \mathbb{E}_{q(\phi)}[z_i])(\sigma^2 \lambda + \mathbb{E}_{q(\phi)}[z_i z_i^T])^{-1}
\end{aligned}$$

The above W refers to its value according to the contribution of the i^{th} example. To obtain W, we would have to sum up all n examples.

Since we need $q(\theta)$ to match $\ln p(z_i|x_i, W)$ so we have $q(\theta)$ set as $p(z_i|x_i, W)$. The corresponding distribution is:

$$\begin{aligned}
p(z_i|x_i, W) &\propto p(z_i, x_i, W) \\
&\propto \exp(-\frac{1}{2} z_i^T z_i - \frac{1}{2\sigma^2}(x_i - W z_i)^T (x_i - W z_i) - \frac{\lambda}{2} W^T W) \\
&\propto \exp(-\frac{1}{2\sigma^2}(\sigma^2 z_i^T z_i + x_i^T x_i - 2 x_i^T W z_i + z_i^T W^T W z_i)) \\
&\propto \exp(-\frac{1}{2\sigma^2}(z_i^T \underbrace{(W^T W + \sigma^2 I)}_{M} z_i - 2 z_i^T W^T x_i)) \\
&\propto \exp(-\frac{1}{2\sigma^2}(z_i^T M z_i - 2 z_i^T M M^{-1} W^T x_i + (M^{-1} W^T x_i)^T (M^{-1} W^T x_i))) \\
&\propto \exp(-\frac{1}{2}(z_i - M^{-1} W^T x_i)^T (\sigma^{-2} M)(z_i - M^{-1} W^T x_i)) \\
&= \mathcal{N}(M^{-1} W^T x_i, \sigma^2 M^{-1})
\end{aligned}$$

Hence $\mathbb{E}_{p(z_i|x_i)}[z_i] = M^{-1} W^T x_i$ and $\mathbb{E}_{p(z_i|x_i)}[z_i z_i^T] = \sigma^2 M^{-1} - \mathbb{E}_{p(z_i|x_i)}[z_i] \mathbb{E}_{p(z_i|x_i)}[z_i^T]$.

An EM algorithm for Bayesian PCA

1. Initialise $W \sim N(0, \lambda^{-1})$ and $z_{1:n} \sim N(0, I)$.
2. For iteration $t = 1, ..., T$:
   (a) E-Step: Compute the following where $M = W^T W + \sigma^2 I$
   $\mathbb{E}_{p(z_{1:n}|p(x_{1:n})}[z_{1:n}] = \sum_{i=1}^n M^{-1} W^T x_i$
   $\mathbb{E}_{p(z_{1:n}|p(x_{1:n})}[z_{1:n} z_{1:n}^T] = \sum_{i=1}^n \sigma^2 M^{-1} - \mathbb{E}_{p(z_i|x_i)}[z_i] \mathbb{E}_{p(z_i|x_i)}[z_i^T]$
   (b) M-Step: Update W with the calculated values above with
   $W = \sum_{i=1}^n (x_i^T \mathbb{E}_{q(\phi)}[z_i])(\sigma^2 \lambda + \mathbb{E}_{q(\phi)}[z_i z_i^T])^{-1}$
   (c) Calculate $\ln p(x, W, z)$ using equation
   $\ln p(x, W, z) = \sum_{i=1}^n \frac{dk}{2} \ln \frac{\lambda}{2\pi} - \frac{\lambda}{2} tr(W^T W) + \frac{d}{2} \ln \frac{1}{2\pi\sigma^2} - \frac{1}{2\sigma^2}(x_i - W z_i)^T (x_i - W z_i) + \frac{k}{2} \ln \frac{1}{2\pi} - \frac{1}{2} z_i^T z_i$

In [1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
from scipy.io import loadmat
from scipy.stats import norm
```

In [2]:
```
data = loadmat('hw2_data_mat/mnist_mat.mat')
```

In [3]:
```
sigma = 1.5
lamda = 1
T = 100
```

In [4]:
```
dim = data['Xtrain'].shape[0]
n = data['Xtrain'].shape[1]
print dim, n
```

```
15 11791
```

In [5]:
```
Xtrain_P = []
Xtrain_N = []
```

In [6]:
```
for i in xrange(n):
    if data['ytrain'][:, i][0] == 1:
        Xtrain_P.append(data['Xtrain'][:, i])
    else:
        Xtrain_N.append(data['Xtrain'][:, i])
```

In [7]:
```
Xtrain_P = np.transpose(np.array(Xtrain_P))
Xtrain_N = np.transpose(np.array(Xtrain_N))
```

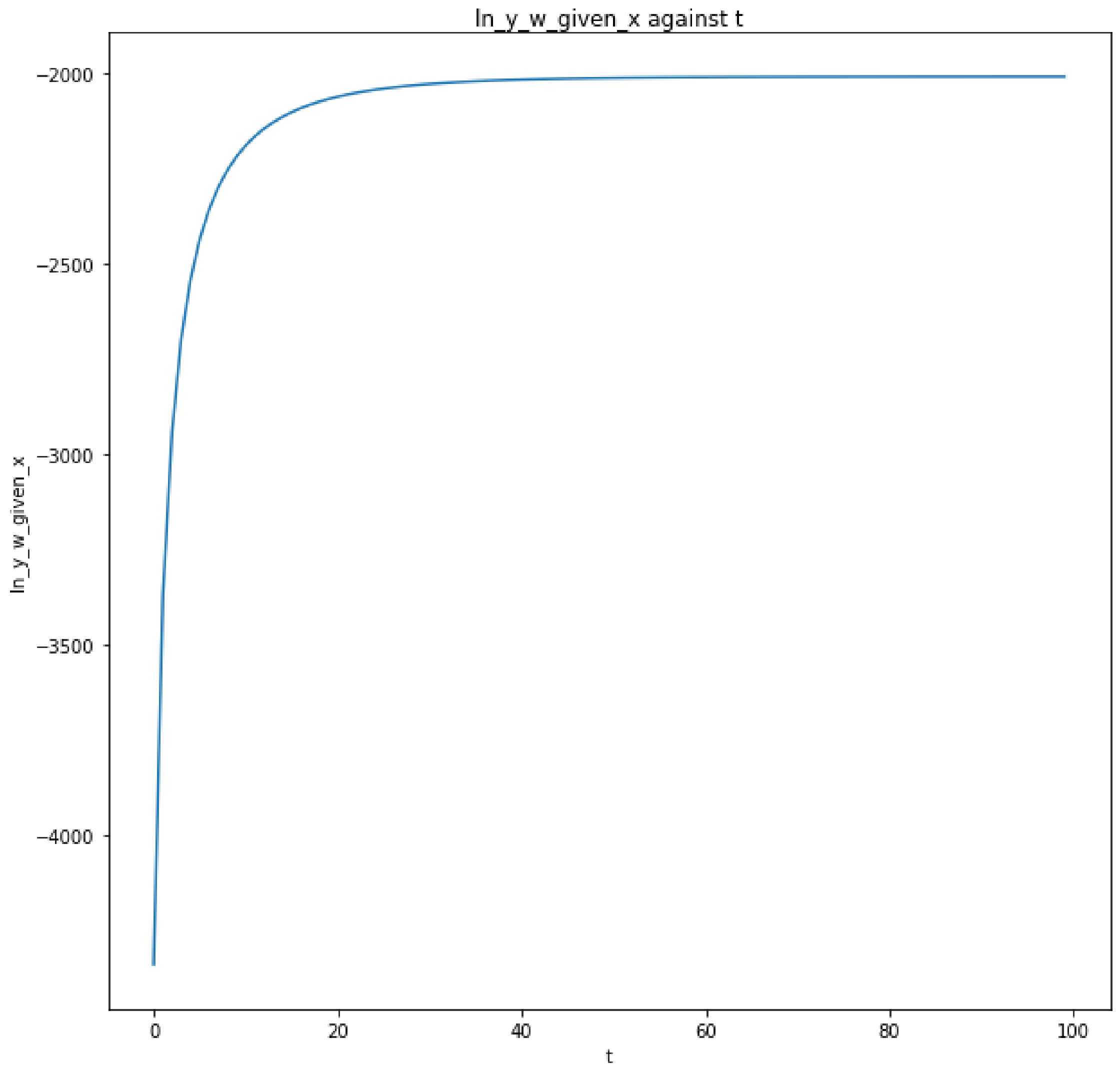
In [8]:
```
n_P = Xtrain_P.shape
n_N = Xtrain_N.shape
```

In [9]:
```
w = np.zeros((15, 1))
w_t = []
ln = []
for i in range(T):
    # E-Step
    X_t_w_P = np.dot(Xtrain_P.T, w)
    X_t_w_N = np.dot(Xtrain_N.T, w)
    E_phi_P = X_t_w_P + sigma * norm.pdf(-X_t_w_P / sigma) / (1 - norm.c
df(-X_t_w_P / sigma))
    E_phi_N = X_t_w_N + sigma * - norm.pdf(-X_t_w_N / sigma) /
norm.cdf(-X_t_w_N / sigma)
    # M-Step
    x_x_T = np.divide((np.dot(Xtrain_P, Xtrain_P.T) + np.dot(Xtrain_N, X
train_N.T)), sigma**2)
    inverted = np.linalg.inv(np.dot(np.identity(15), lamda) + x_x_T)
    x_E_phi = np.divide((np.dot(Xtrain_P, E_phi_P) + np.dot(Xtrain_N, E_
phi_N)), sigma **2)
    w = np.dot(inverted, x_E_phi)
    # Calculate ln
    ln_y_w_given_x = (dim / 2.0 * np.log(lamda/(2 * np.pi)) - lamda / 2.
0 * np.dot(w.T, w) +
                      np.sum(np.log(norm.cdf(np.dot(Xtrain_P.T,
w)/sigma))) +
                      np.sum(np.log(1 - norm.cdf(np.dot(Xtrain_N.T, w)/si
gma))))

    if i in [0, 4, 9, 24, 49, 99]:
        w_t.append(w)
    ln.append(ln_y_w_given_x[0][0])
```

In [10]:
```
x = range(100)
plt.figure(figsize=(10, 10))
plt.plot(x, ln)
plt.ylabel('ln_y_w_given_x')
plt.xlabel('t')
plt.title('ln_y_w_given_x against t')
```

Out[10]: `Text(0.5,1,u'ln_y_w_given_x against t')`



In [11]:
```
pred_Y = []
true_P = 0
true_N = 0
false_P = 0
false_N = 0
print data['ytest'].shape
```

```
(1, 1991)
```

In [12]:
```
for i in xrange(data['Xtest'].shape[1]):
    prob_P = norm.cdf(np.dot(data['Xtest'][:, i].reshape((15,1)).T, w_t[5

    if prob_P >= 0.5 and data['ytest'][:, i][0] == 1:
        true_P += 1
    elif prob_P >= 0.5 and data['ytest'][:, i][0] == 0:
        false_P += 1
        print i, prob_P
    elif prob_P < 0.5 and data['ytest'][:, i][0] == 1:
        false_N += 1
        print i, prob_P[0][0]
    else:
        true_N += 1
    pred_Y.append(prob_P[0][0])
```

```
40 [[ 0.75483351]]
46 [[ 0.78198279]]
64 [[ 0.9757452]]
74 [[ 0.93095418]]
80 [[ 0.99911947]]
81 [[ 0.5119825]]
84 [[ 0.76368115]]
94 [[ 0.57854307]]
138 [[ 0.95722124]]
142 [[ 0.8313627]]
156 [[ 0.93107843]]
162 [[ 0.7001229]]
163 [[ 0.80849723]]
183 [[ 0.55764663]]
195 [[ 0.56546922]]
210 [[ 0.50613315]]
221 [[ 0.99995144]]
223 [[ 0.92184408]]
231 [[ 0.97702347]]
239 [[ 0.69336726]]
259 [[ 0.80182825]]
263 [[ 0.98936797]]
269 [[ 0.57763019]]
271 [[ 0.79579723]]
293 [[ 0.65842084]]
301 [[ 0.86329532]]
312 [[ 0.88172473]]
340 [[ 0.50463717]]
348 [[ 0.84173697]]
357 [[ 0.6722796]]
360 [[ 0.78870087]]
396 [[ 0.83967772]]
420 [[ 0.84403436]]
440 [[ 0.68536092]]
441 [[ 0.7419132]]
465 [[ 0.99140899]]
476 [[ 0.63800794]]
489 [[ 0.92204447]]
529 [[ 0.55398608]]
559 [[ 0.55595442]]
564 [[ 0.62612537]]
586 [[ 0.5002525]]
587 [[ 0.84138435]]
592 [[ 0.72139615]]
603 [[ 0.71099252]]
676 [[ 0.53796456]]
715 [[ 0.55112567]]
730 [[ 0.57909342]]
744 [[ 0.56129254]]
832 [[ 0.77852964]]
842 [[ 0.99887344]]
909 [[ 0.64406079]]
988 0.474083033582
1002 0.269198185046
1010 0.453734536313
1038 0.303886863706
1094 0.0757977206357
```

1097 0.432676699926
1117 0.200664111189
1140 0.144547024545
1149 0.242954813462
1168 0.00737809598847
1181 0.386164722497
1184 0.0181905757155
1201 0.329625700235
1253 0.47781449629
1293 0.43736224498
1327 0.0444890217592
1342 0.0325323547972
1346 0.371389733048
1351 0.0513670697336
1370 0.0474757622353
1373 0.400084382476
1382 3.52883864409e-05
1390 0.085294317303
1392 0.0271206625297
1407 0.181200072876
1424 0.337774248592
1427 0.379945440526
1429 0.220657060203
1435 0.259888936895
1445 0.477676387607
1446 0.475532410749
1459 0.483528548889
1463 0.400709456576
1467 0.30159094869
1475 0.0544963824207
1477 0.174858993188
1482 0.00422627610849
1484 0.412956839443
1491 0.0644173987111
1496 0.245623856429
1502 0.131667521065
1504 0.0444568111736
1516 0.045001171609
1519 0.0141013720126
1567 0.462978305496
1618 0.289568538565
1619 0.457926185862
1653 0.0309947503052
1655 0.297382340981
1658 0.467922433693
1660 0.343703062665
1662 0.0265922674236
1674 0.219925031864
1678 0.391164092904
1681 0.295885693414
1688 0.344571402597
1707 0.34317243534
1708 0.342637632353
1757 0.160389173707
1758 0.0531235773196
1837 0.384270370656
1842 0.441564104377

```
1901 0.397782616551
1902 0.0109345042581
1919 0.0962604432108
1924 0.445970034101
1958 0.00868376514755
1971 0.293557212774
1972 0.0741919661533
1973 0.474485374353
1977 0.00011529151587
1980 0.37273573069
1981 0.016122208355
1982 0.0109516727919
1983 0.221224009317
1986 0.437457475958
1988 0.410013424395
```

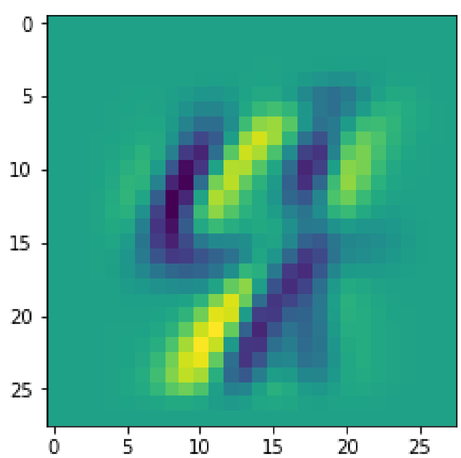
In [13]:
```
print true_P, true_N, false_P, false_N
```

```
932 930 52 77
```

In [14]:
```
Q = data['Q']
```

In [15]:
```
misclass_1 = np.dot(Q, data['Xtest'][:, 221]).reshape(28, 28)
plt.imshow(misclass_1)
print pred_Y[221]
```

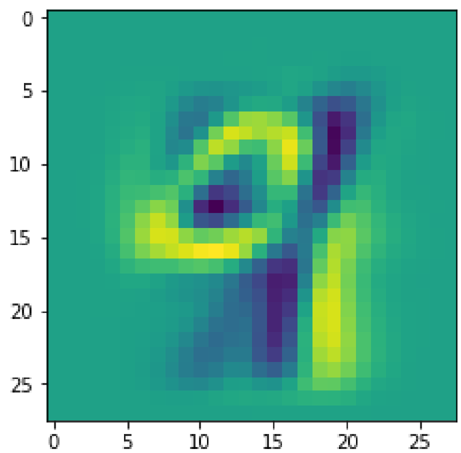
```
0.999951444623
```



In [16]:

```
misclass_1 = np.dot(Q, data['Xtest'][:, 263]).reshape(28, 28)
plt.imshow(misclass_1)
print pred_Y[263]
```

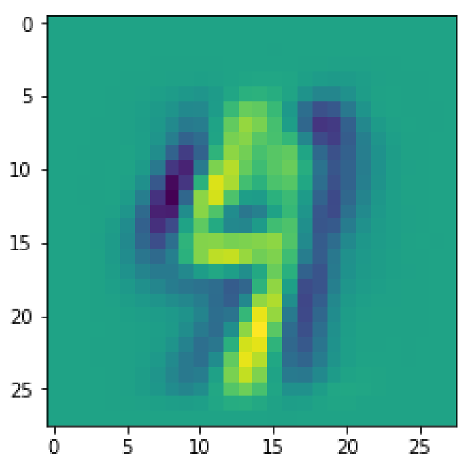
```
0.989367967151
```



In [17]:

```
misclass_1 = np.dot(Q, data['Xtest'][:, 269]).reshape(28, 28)
plt.imshow(misclass_1)
print pred_Y[269]
```

```
0.577630188348
```



In [18]:

```
abs_diff = np.abs(np.array(pred_Y) - 0.5)
```

In [19]:

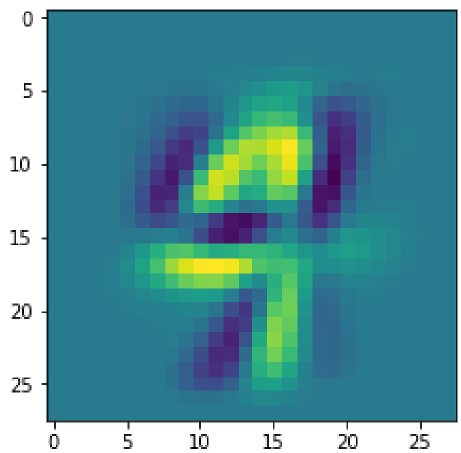
```
ambi_index = np.argsort(abs_diff)[:3]
ambi_index
```

Out[19]:

```
array([586, 340, 210])
```

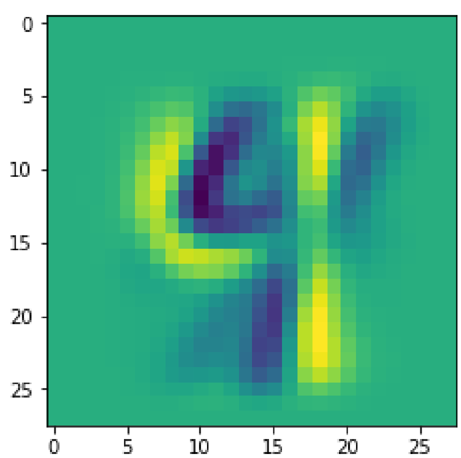
In [20]:
```
ambi_1 = np.dot(Q, data['Xtest'][:, ambi_index[0]]).reshape(28, 28)
plt.imshow(ambi_1)
print pred_Y[ambi_index[0]]
```

```
0.500252498451
```



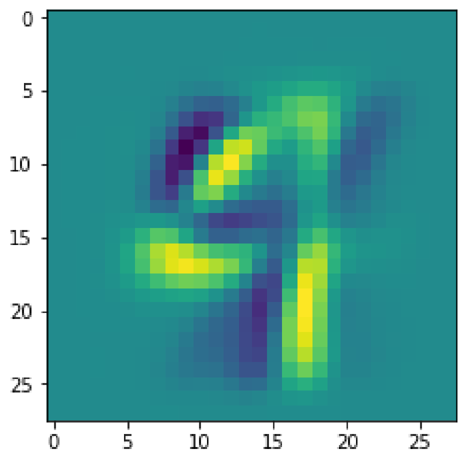
In [21]:
```
ambi_2 = np.dot(Q, data['Xtest'][:, ambi_index[1]]).reshape(28, 28)
plt.imshow(ambi_2)
print pred_Y[ambi_index[1]]
```

```
0.504637167196
```



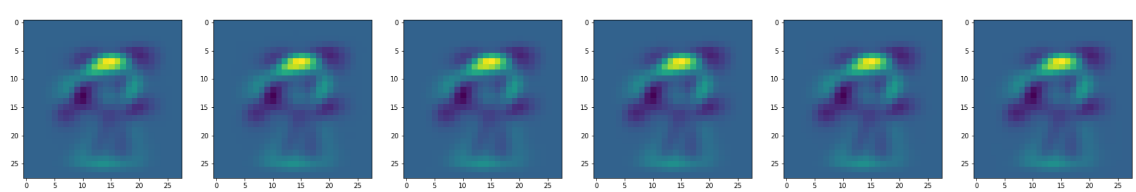
In [22]:
```
ambi_3 = np.dot(Q, data['Xtest'][:, ambi_index[2]]).reshape(28, 28)
plt.imshow(ambi_3)
print pred_Y[ambi_index[2]]
```

```
0.506133148466
```



In [23]:
```
plt.figure(figsize=(30, 180))
ax1 = plt.subplot(161)
plt.imshow(np.dot(Q, w_t[0]).reshape(28, 28))
ax2 = plt.subplot(162)
plt.imshow(np.dot(Q, w_t[1]).reshape(28, 28))
ax3 = plt.subplot(163)
plt.imshow(np.dot(Q, w_t[2]).reshape(28, 28))
ax4 = plt.subplot(164)
plt.imshow(np.dot(Q, w_t[3]).reshape(28, 28))
ax5 = plt.subplot(165)
plt.imshow(np.dot(Q, w_t[4]).reshape(28, 28))
ax6 = plt.subplot(166)
plt.imshow(np.dot(Q, w_t[5]).reshape(28, 28))
```

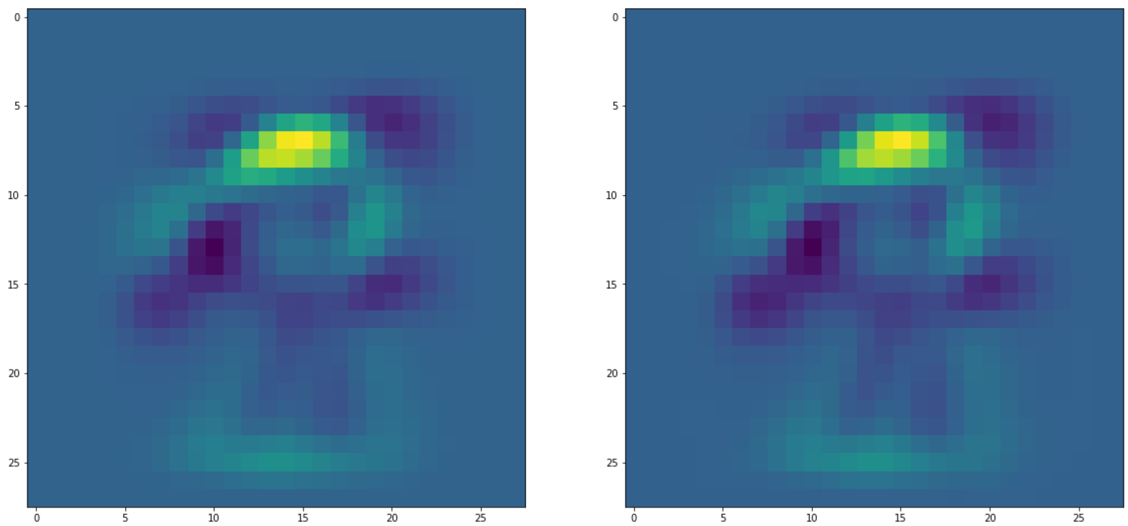
Out[23]: <matplotlib.image.AxesImage at 0x10d421c10>



The images all look quite similar, but if we compare the first and last Ws closely, we can observe slightly more structure in the form of more defined patches of yellow and blue highlighting characteristics of 9 and 4 respectively.

In [24]:
```
plt.figure(figsize=(20, 40))
ax1 = plt.subplot(121)
plt.imshow(np.dot(Q, w_t[0]).reshape(28, 28))
ax2 = plt.subplot(122)
plt.imshow(np.dot(Q, w_t[5]).reshape(28, 28))
```

Out[24]: <matplotlib.image.AxesImage at 0x114329750>



In [ ]: