Bayesian Models for Machine Learning
Problem Set #1

Si Kai Lee sl3950@columbia.edu

September 30, 2016

Problem 1

Let X_i be the event where door i is chosen, D_i be the event where the prize is behind door i and Y_i be the event where door i is opened by the host. Assume $X = X_1$ and $Y = Y_3$.

$$\begin{aligned}
P(D_1|X_1, Y_3) &= \frac{P(D_1, X_1, Y_3)}{P(X_1, Y_3)} \\
&= \frac{P(D_1, X_1, Y_3)}{P(D_1, X_1, Y_3) + P(D_2, X_1, Y_3)} \\
&= \frac{1/3 * 1/2}{1/3 * 1/2 + 1/3 * 1} \text{as the host cannot open the selected door } P(Y_3|X_1, D_2) = 1 \\
&= 1/3
\end{aligned}$$

Since $P(D_3|X_1, Y_3) = 0$ as the host opened the door to show that it is empty, so $P(D_2|X_1, Y_3) = 1 - 1/3 = 2/3$. Hence the friend should switch.

Problem 2

The conjugate prior is the Dirichlet distribution which is defined as $\frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \prod_{i=1}^k x_i^{\alpha_i - 1}$.

Likelihood: $X_i \sim Mult(\pi)$, Prior: $\pi \sim Dir(\alpha)$. Assume $\forall a_i = \alpha$.

$$\begin{aligned}
p(\pi|X) &= \prod_{j=1}^{N} \frac{\Gamma(\sum_i x_i + 1)}{\prod_i \Gamma(x_i + 1)} \prod_{i=1}^{K} \pi_i^{x_j} \cdot \frac{\Gamma\left(\sum_K \alpha\right)}{\prod_K \Gamma(\alpha)} \prod_{i=1}^{K} \pi_i^{\alpha - 1} \\
&\propto \prod_{j=1}^{N} \prod_{j=1}^{K} \pi_i^{x_j} \cdot \prod_{i=1}^{K} \pi_i^{\alpha-1} = \prod_{i=1}^{K} \pi_i^{\alpha + n_i - 1} \text{ where } n_i \text{ is the \# of observations in the ith category} \\
&= \frac{\Gamma\left(\sum_{i=1}^{K} \alpha + n_i\right)}{\prod_{i=1}^{K} \Gamma(\alpha + n_i)} \prod_{i=1}^{K} \pi_i^{\alpha + n_i - 1} = Dir(\alpha + n)
\end{aligned}$$

The parameter α_i for the ith class is shifted positively by the number of observations n_i in the same category.

Problem 3

a

We note that $p(\mu, \lambda|X) \propto p(X|\mu, \lambda)p(\mu|\lambda)p(\lambda|b, c)$. Expand $p(x|\mu, \lambda)$ and remove non-μ, λ terms:

$$p(X|\mu, \lambda) = \prod_{i=1}^{N} \left(\frac{\lambda}{2\pi}\right)^{1/2} \exp\left(-\frac{\lambda}{2}(x_i - \mu)^2\right) \propto \lambda^{N/2} \exp\left(-\frac{\lambda}{2}\sum_{i=1}^{N}(x_i - \mu)^2\right)$$

Next, expand $p(\mu, \lambda)$ and remove non-μ, λ terms:

$$\begin{aligned} p(\mu, \lambda) = p(\mu|\lambda)p(\lambda|b, c) &= \left(\frac{\lambda}{2\pi a}\right)^{1/2} \exp\left(-\frac{\lambda}{2a}\mu^2\right) \cdot \frac{c^b}{\Gamma(b)}\lambda^{b-1}\exp(-c\lambda) \\ &\propto \lambda^{b-1/2} \exp\left(-\frac{\lambda}{2a}\mu^2 - c\lambda\right) \end{aligned}$$

Due to conjugacy, we know $p(\mu, \lambda|x)$ is a Normal Gamma distribution:

$$\begin{aligned} p(\mu, \lambda|X) &\propto \lambda^{N/2} \exp\left(-\frac{\lambda}{2}\sum_{i=1}^{N} x_i^2 + \lambda\mu\sum_{i=1}^{N} x_i - \frac{N\lambda\mu^2}{2}\right) \cdot \lambda^{b-1/2} \exp\left(-\frac{\lambda}{2a}\mu^2 - c\lambda\right) \\ &= N(\mu_n, \lambda_n)\ Gamma(\alpha_n, \beta_n) \end{aligned}$$

Integrating w.r.t. to λ i.e. removing all λ terms, we obtain the distribution of μ:

$$\begin{aligned} p(\mu|\lambda, X) &\propto \exp\left(\lambda\mu\sum_{i=1}^{N} x_i - \frac{N\lambda\mu^2}{2} - \frac{\lambda}{2a}\mu^2\right) \\ &\propto \exp\left[-\frac{\lambda}{2a}\left((aN+1)\mu^2 - 2aN\bar{x}\mu\right)\right] \text{ where } \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \\ &\propto \exp\left[-\frac{\lambda}{2a}\left((aN+1)\mu^2 - 2aN\bar{x}\mu + \frac{(aN\bar{x})^2}{aN+1}\right)\right] \\ &\propto \exp\left[-\frac{\lambda(aN+1)}{2a}\left(\mu^2 - 2\frac{aN\bar{x}}{aN+1}\mu + \frac{aN\bar{x}}{aN+1}^2\right)\right] \\ &\propto \lambda^{1/2} \exp\left[-\frac{\lambda_n}{2}(\mu - \mu_n)^2\right] \text{ where } \mu_n = \frac{aN\bar{x}}{aN+1}, \lambda_n = \frac{\lambda}{a}(aN+1) \\ &= N(\mu_n, \lambda_n) \end{aligned}$$

Comparing terms of $p(\mu, \lambda|X)$ and $p(\mu|\lambda, X)$, we find $p(\lambda|X)$:

$$\begin{aligned} p(\lambda|X) &\propto \lambda^{N/2+b-1} \exp\left(-\lambda\left(\frac{1}{2}\sum_{i=1}^{N} x_i^2 + c - \frac{1}{2}\frac{(aN\bar{x})^2}{a(aN+1)}\right)\right) \\ &= Gamma(\alpha_n, \beta_n) \text{ where } \alpha_n = b + N/2,\ \beta_n = c + \frac{1}{2}\sum_{i=1}^{N} x_i^2 - \frac{1}{2}\frac{(aN\bar{x})^2}{a(aN+1)} \end{aligned}$$

b

Let $\kappa_n = \frac{1}{a}(aN+1)$.

$$
\begin{aligned}
&p(x^*|x_{1:n}) \\
&= \int_0^\infty \int_{-\infty}^\infty p(x^*|\mu,\lambda)p(\mu,\lambda|x_{1:n})d\mu d\lambda \\
&= \int_0^\infty \int_{-\infty}^\infty \left(\frac{\lambda}{2\pi}\right)^{1/2} \exp\left[-\frac{\lambda}{2}(x^*-\mu)^2\right] \left(\frac{\kappa_n\lambda}{2\pi}\right)^{1/2} \exp\left[-\frac{\kappa_n\lambda}{2}(\mu-\mu_n)^2\right] \frac{\beta_n^{\alpha_n}}{\Gamma(\alpha_n)}\lambda^{\alpha_n-1}\exp[-\beta_n\lambda]d\mu d\lambda \\
&= \int_0^\infty \frac{\kappa_n\lambda^{\alpha_n}}{2\pi}\frac{\beta_n^{\alpha_n}}{\Gamma(\alpha_n)} \exp\left[-\beta_n\lambda - \frac{\lambda}{2}x^{*2} - \frac{\lambda}{2}\mu_n^2\right] \int_{-\infty}^\infty \exp\left[-\frac{\lambda}{2}((d+1)\mu^2 - 2(x^*+\kappa_n\mu_n)\mu)\right] d\mu d\lambda \\
&= \int_0^\infty Z \int_{-\infty}^\infty \exp\left[-\frac{\lambda(\kappa_n+1)}{2}\left(\mu - \frac{x^*+\kappa_n\mu_n}{\kappa_n+1}\right)^2 + \frac{\lambda}{2}\frac{(x^*+\kappa_n\mu_n)^2}{\kappa_n+1}\right] d\mu d\lambda \text{ where } Z \text{ is constant w.r.t. } \mu \\
&= \int_0^\infty Z\exp\left[\frac{\lambda}{2}\frac{(x^*+\kappa_n\mu_n)^2}{\kappa_n+1}\right] \int_{-\infty}^\infty \exp\left[-\left(\mu-\frac{x^*+\kappa_n\mu_n}{\kappa_n+1}\right)^2 / \frac{2}{\lambda(\kappa_n+1)}\right] d\mu d\lambda \\
&= \int_0^\infty Z\exp\left[\frac{\lambda}{2}\frac{(x^*+\kappa_n\mu_n)^2}{\kappa_n+1}\right] \left(\frac{2\pi}{(\kappa_n+1)\lambda}\right)^{1/2} d\lambda \\
&= \int_0^\infty \left(\frac{\kappa_n}{\kappa_n+1}\right)^{1/2}(2\pi)^{-1/2}\frac{\beta_n^{\alpha_n}}{\Gamma(\alpha_n)}\lambda^{\alpha_n-1/2}\exp\left[-\beta_n\lambda - \frac{\lambda}{2}x^{*2} - \frac{\lambda}{2}\mu_n^2 + \frac{\lambda}{2}\frac{(x^*+\kappa_n\mu_n)^2}{\kappa_n+1}\right] d\lambda \\
&= \int_0^\infty \left(\frac{\kappa_n}{\kappa_n+1}\right)^{1/2}(2\pi)^{-1/2}\frac{\beta_n^{\alpha_n}}{\Gamma(\alpha_n)}\lambda^{\alpha_n-1/2}\exp\left[\frac{\lambda}{2(\kappa_n+1)}(\kappa_n x^{*2} + \kappa_n\mu_n^2 + 2\beta_n(\kappa_n+1) - 2\kappa_n\mu_n x^*)\right] d\lambda \\
&= Z'\frac{\Gamma(\alpha_{n+1})}{\beta_{n+1}^{\alpha_{n+1}}}\int_0^\infty \frac{\beta_{n+1}^{\alpha_{n+1}}}{\Gamma(\alpha_{n+1})}\lambda^{\alpha_{n+1}}\exp(-\beta_{n+1}\lambda)d\lambda \text{ where } Z' \text{ is constant w.r.t. } \lambda \text{ and see below for } \alpha_{n+1}, \beta_{n\cdot} \\
&= \left(\frac{\kappa_n}{\kappa_n+1}\right)^{1/2}(2\pi)^{-1/2}\frac{\Gamma(\alpha_{n+1})}{\Gamma(\alpha_n)}\frac{\beta_n^{\alpha_n}}{\beta_{n+1}^{\alpha_{n+1}}}
\end{aligned}
$$

We know that the above corresponds to a t-distribution according to Murphy[1] and

$$
\begin{aligned}
\alpha_{n+1} &= \alpha_n + 1/2 \\
\beta_{n+1} &= \beta_n + \frac{\kappa_n}{2(\kappa_n+1)}(x^*-\mu_n)^2
\end{aligned}
$$

[1] Murphy, Kevin P. "Conjugate Bayesian analysis of the Gaussian distribution." def 1, no. 2?2 (2007): 16.

HW1 Problem 4

September 28, 2016

```
In [1]: import matplotlib.pyplot as plt
        %matplotlib inline

In [2]: import numpy as np

In [3]: from scipy.io import loadmat
        from scipy.special import gammaln

In [4]: data = loadmat('hw1_data_mat/mnist_mat.mat')

In [5]: a = b = c = e = f = 1.0

In [6]: dim = data['Xtrain'].shape[0]
        n = data['Xtrain'].shape[1]

In [7]: Xtrain_P = []
        Xtrain_N = []

In [8]: 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 [9]: Xtrain_P = np.transpose(np.array(Xtrain_P))
        Xtrain_N = np.transpose(np.array(Xtrain_N))

In [10]: n_P = Xtrain_P.shape[1]
         n_N = Xtrain_N.shape[1]

In [11]: x_bar_P = []
         x_bar_N = []
         for i in xrange(15):
             x_bar_P.append(np.mean(Xtrain_P[i]))
             x_bar_N.append(np.mean(Xtrain_N[i]))

In [12]: x_bar_P, x_bar_N
```

```
Out[12]: ([-0.47061730664131385,
           0.58286614230845868,
           0.20587045510959476,
           -0.40882726075281095,
           0.56292712801894862,
           0.46453685788723231,
           0.083284904986044864,
           -0.16829601501539504,
           0.3337881746038015,
           -0.050308009077472522,
           -0.1021845627499352,
           0.11035410083574052,
           -0.046616043393231263,
           -0.049521882310979128,
           0.030770746331257301],
          [0.44559820987050736,
           -0.63681884607506056,
           -0.275012249244438 4,
           0.40989926135792615,
           -0.57246165678908867,
           -0.45722997130591531,
           -0.080959618408045581,
           0.17310278209683494,
           -0.33627257361027529,
           0.052285728424823333,
           0.12688412801189222,
           -0.10852735709995273,
           0.053593082195101144,
           0.052877011940705786,
           -0.030761838621929601])
```

```python
In [13]: print Xtrain_P.shape

         x_2_P = []
         x_2_N = []
         for i in xrange(15):
             x_2_P.append(sum(map(lambda x:x * x, Xtrain_P[i, :])))
             x_2_N.append(sum(map(lambda x:x * x, Xtrain_N[i, :])))
```

```
(15, 5949)
```

```python
In [14]: x_2_P, x_2_N
```

```
Out[14]: ([35736.495824099024,
           21940.399070690841,
           17726.881231744133,
           11416.856006645525,
           9080.5320089872457,
```

```
  8339.6941494976909,
  7658.3117853378553,
  7164.0182013244539,
  6906.9606531207564,
  5776.0189943804389,
  4561.9198564108001,
  4789.244744705602,
  4379.0394427675983,
  5323.9431311741964,
  3699.2630534896903],
 [27725.4275292926,
  22413.030576790454,
  15366.956765373496,
  12264.395076509658,
  11593.133112525949,
  11048.704541648358,
  9373.2972086597674,
  8773.5387732982854,
  6407.3255050389062,
  7003.2779376883327,
  6650.3368683887084,
  4668.7243042215305,
  4665.0021636265847,
  3261.32490157961,
  3750.1433414824633])
```

```
In [15]: mu_n_P = map(lambda x: (a * n_P * x) / (n_P + 1), x_bar_P)
         mu_n_N = map(lambda x: (a * n_N * x) / (n_N + 1), x_bar_N)
```

```
In [16]: kappa_n_P = (a * n_P + 1) / a
         kappa_n_N = (a * n_N + 1) / a
```

```
In [17]: alpha_n_P = b + n_P * 0.5
         alpha_n_N = b + n_N * 0.5
```

```
In [18]: beta_n_P = []
         beta_n_N = []
         for i in xrange(15):
             beta_n_P.append(c + 0.5 * x_2_P[i] - 0.5 * ((a * n_P * x_bar_P[i])
             beta_n_N.append(c + 0.5 * x_2_N[i] - 0.5 * ((a * n_N * x_bar_N[i])
```

```
In [19]: beta_n_P, beta_n_N
```

```
Out[19]: ([17210.564442388786,
  9960.8337436838883,
  8738.3946282008674,
  5212.354434830967,
  3598.8441842234333,
  3529.0742364083558,
```

```
            3809.5271116507975,
            3498.7748645373322,
            3123.1334588649365,
            2881.482612935733,
            2250.9063565412357,
            2359.4049173324956,
            2184.0570541393131,
            2655.6780777356635,
            1847.8156279855054],
           [13283.825796024681,
            10023.14081583915,
            7463.595887621439,
            5642.5026941465539,
            4840.4826144439412,
            4914.7947230627387,
            4668.5063038857888,
            4300.2578481429955,
            2874.4148114346649,
            3494.65491331254,
            3279.1496037569946,
            2300.9639552710833,
            2325.1127675575008,
            1623.4967956573598,
            1873.3080286259246])
```

```
In [20]: p_y_1_y_star = (e + n_P)/(n + e + f)
```

```
In [21]: p_y_0_y_star = (f + n_N)/(n + e + f)
```

```
In [22]: p_y_1_y_star, p_y_0_y_star
```

```
Out[22]: (0.5045365895022471, 0.4954634104977529)
```

```
In [23]: alpha_n_P_1 = alpha_n_P + 0.5
         alpha_n_N_1 = alpha_n_N + 0.5
```

```
In [24]: log_kappa_t_P = np.log((kappa_n_P / (kappa_n_P + 1)) ** 0.5)
         log_kappa_t_N = np.log((kappa_n_N / (kappa_n_N + 1)) ** 0.5)
```

```
In [25]: log_pi_t = np.log((2 * np.pi) ** -0.5)
```

```
In [26]: log_gamma_t_P = gammaln(alpha_n_P + 0.5) - gammaln(alpha_n_P)
         log_gamma_t_N = gammaln(alpha_n_N + 0.5) - gammaln(alpha_n_N)
```

```
In [27]: log_beta_alpha_n_P = map(lambda x: alpha_n_P * np.log(x), beta_n_P)
         log_beta_alpha_n_N = map(lambda x: alpha_n_N * np.log(x), beta_n_N)
```

```
In [28]: pred_Y = []
         true_P = 0
```

```
true_N = 0
false_P = 0
false_N = 0
print data['ytest'].shape
```

```
(1, 1991)
```

```
In [29]: for i in xrange(data['Xtest'].shape[1]):
             log_beta_alpha_n_P_1 = alpha_n_P_1 * np.log(beta_n_P + kappa_n_P /
             log_beta_alpha_n_N_1 = alpha_n_N_1 * np.log(beta_n_N + kappa_n_N /
             log_x_star_y_star_P = 0
             log_x_star_y_star_N = 0
             for j in xrange(dim):
                 log_x_star_y_star_P += log_kappa_t_P + log_pi_t + log_gamma_t_P
                 log_x_star_y_star_N += log_kappa_t_N + log_pi_t + log_gamma_t_N
             prob_P = (np.exp(log_x_star_y_star_P) * p_y_1_y_star)/(np.exp(log_x
             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
             else:
                 true_N += 1
             pred_Y.append(prob_P)
```

```
40 0.735341099219
42 0.519414358248
55 0.726598587938
64 0.701842013806
74 0.700507707503
80 0.881922974112
84 0.697147381991
138 0.687353528054
140 0.51157890678
142 0.595579185123
162 0.633583349415
163 0.544612037314
165 0.599412617182
183 0.730262330533
195 0.680087852783
221 0.869011319805
223 0.609825384577
231 0.924084301304
259 0.720940241961
```

263 0.743521958157
269 0.532529824223
301 0.732475379438
312 0.691490895793
340 0.538750403504
359 0.53902758375
360 0.642289312963
361 0.655817783764
363 0.623808389513
396 0.561092410497
420 0.659919216283
440 0.574170617614
452 0.662089382196
464 0.686901195342
465 0.610660497954
483 0.552351348601
486 0.726901976947
489 0.606340817531
529 0.628567971756
564 0.558361268951
592 0.547354973312
603 0.643990976414
676 0.519077919589
696 0.587439192542
698 0.594264341275
730 0.549834245897
740 0.506914849834
744 0.584898620774
809 0.770441364925
842 0.733327738141
909 0.585400537411
948 0.509748290726
974 0.538432740883
982 0.488061012185
1002 0.459395611182
1033 0.421506542228
1047 0.426510207758
1053 0.333858208763
1074 0.442557085153
1094 0.124032156167
1108 0.480684404771
1117 0.281203967899
1149 0.388839878998
1154 0.474475500766
1168 0.135448164486
1181 0.378561400278
1193 0.355863768104
1201 0.245238453639

1215	0.44888313304
1227	0.474213366977
1253	0.486818617865
1316	0.475110658808
1327	0.208879475306
1342	0.202839107736
1364	0.461907434992
1370	0.27444694741
1373	0.372606665472
1382	0.0208425006864
1390	0.269151613696
1392	0.190727601643
1407	0.331766530256
1410	0.418554731716
1412	0.435891432469
1423	0.374732923964
1429	0.384208231477
1435	0.469404897468
1445	0.234467198756
1446	0.49141707841
1450	0.384277032681
1463	0.202181633727
1475	0.314760058107
1482	0.101806915334
1502	0.451700429112
1504	0.471298143308
1505	0.426618896825
1516	0.301102341479
1519	0.374748307052
1567	0.3772504598
1571	0.454909656883
1647	0.280671011541
1653	0.376796621611
1658	0.481346241638
1661	0.435284314152
1662	0.466148252439
1663	0.424153435047
1669	0.302281955375
1671	0.344266742345
1672	0.443586006186
1673	0.470097616182
1674	0.371686677093
1678	0.279276429663
1680	0.367559782431
1681	0.484068088818
1707	0.355900376749
1708	0.309552073111
1743	0.37938380269

```
1757 0.374560779564
1758 0.301756165547
1818 0.250939409432
1837 0.465601466584
1842 0.357526498319
1901 0.251646581501
1902 0.215273072548
1919 0.233849180984
1922 0.415256045997
1924 0.31944791191
1926 0.397682146311
1958 0.207130240425
1971 0.330333149237
1972 0.299037054181
1977 0.0810884879762
1981 0.371858471927
1982 0.262648381326
1983 0.396717687222
1986 0.304740256968
```

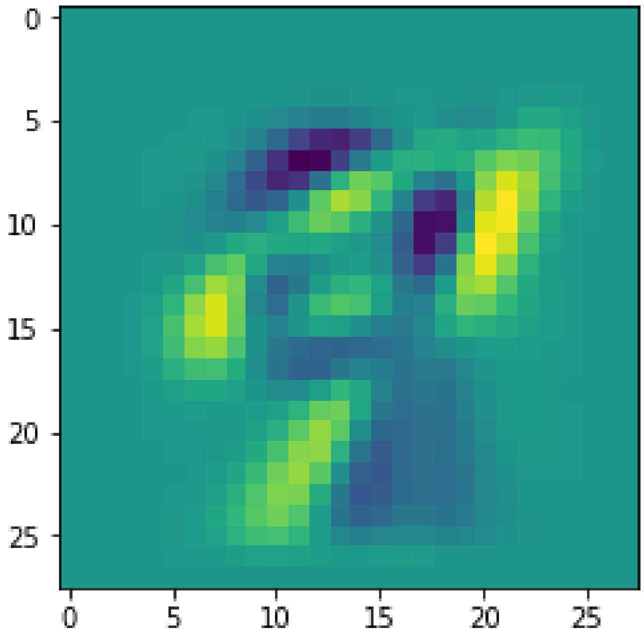
```
In [30]: print true_P, true_N, false_P, false_N
```

```
927 930 52 82
```

```
In [31]: Q = data['Q']
```

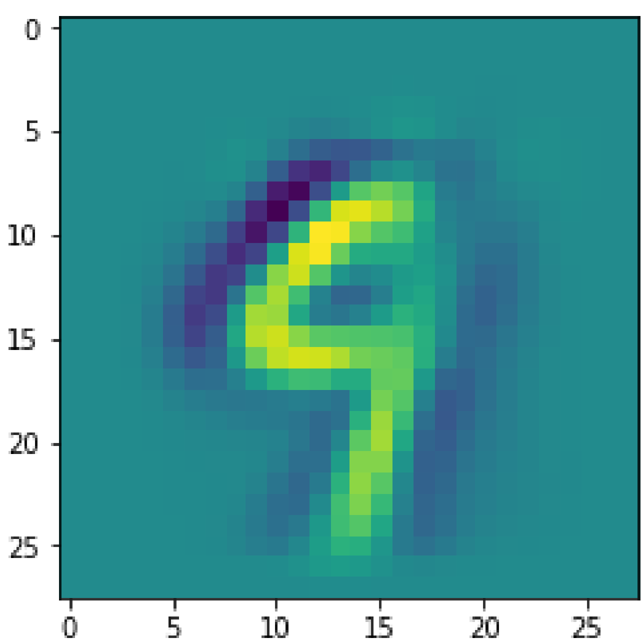
```
In [32]: misclass_1 = np.dot(Q, data['Xtest'][:, 1986]).reshape(28, 28)
         plt.imshow(misclass_1)
         print pred_Y[1986]
```

```
0.304740256968
```



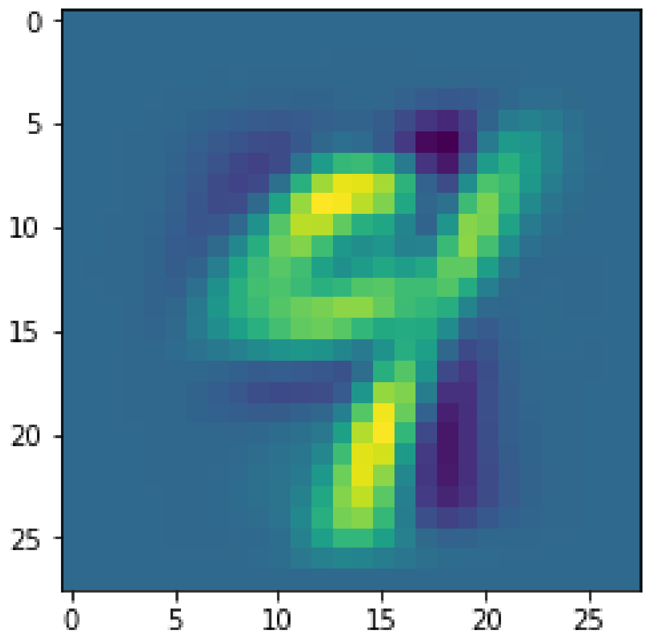

```
In [33]: misclass_2 = np.dot(Q, data['Xtest'][:, 1958]).reshape(28, 28)
         plt.imshow(misclass_2)
         print pred_Y[1958]
```

```
0.207130240425
```



```
In [34]: misclass_3 = np.dot(Q, data['Xtest'][:, 231]).reshape(28, 28)
         plt.imshow(misclass_3)
         print pred_Y[231]
```

```
0.924084301304
```

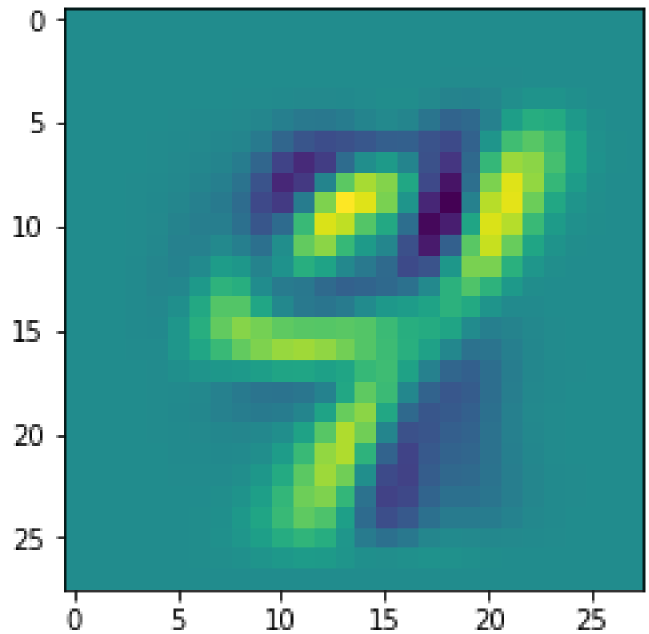



```
In [35]: abs_diff = np.abs(np.array(pred_Y) - 0.5)
```

```
In [36]: ambi_index = np.argsort(abs_diff)[:3]
```

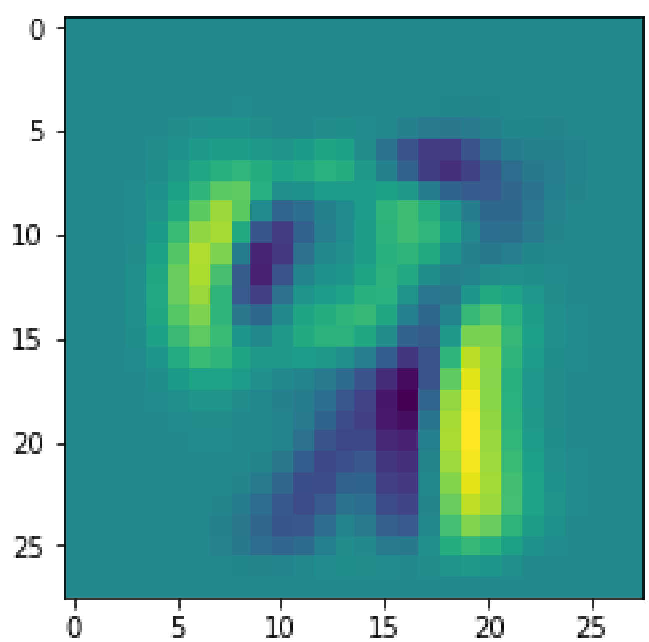
```
In [37]: ambi_1 = np.dot(Q, data['Xtest'][:, ambi_index[0]]).reshape(28, 28)
         plt.imshow(ambi_1)
         print pred_Y[ambi_index[0]]
```

```
0.499616378259
```



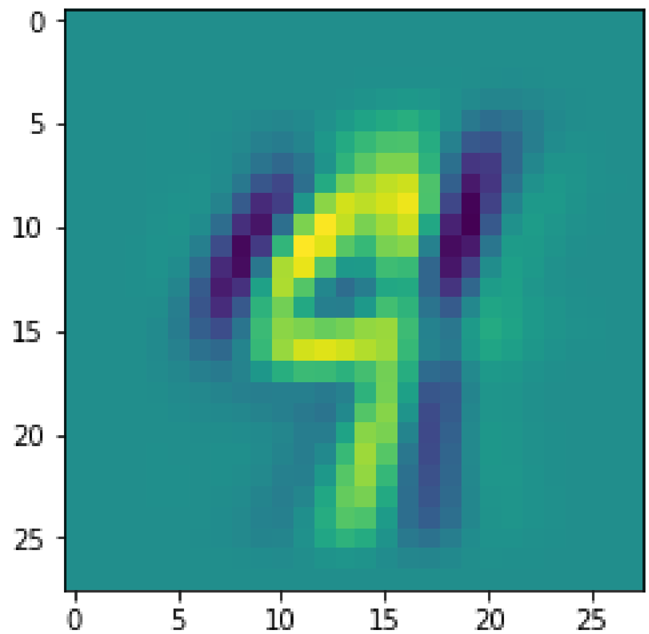

```
In [38]: ambi_2 = np.dot(Q, data['Xtest'][:, ambi_index[1]]).reshape(28, 28)
         plt.imshow(ambi_2)
         print pred_Y[ambi_index[1]]
```

```
0.502011187547
```



```
In [39]: ambi_3 = np.dot(Q, data['Xtest'][:, ambi_index[2]]).reshape(28, 28)
         plt.imshow(ambi_3)
         print pred_Y[ambi_index[2]]
```

```
0.496365983337
```




```
In [ ]:
```