Question 1

c

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
```

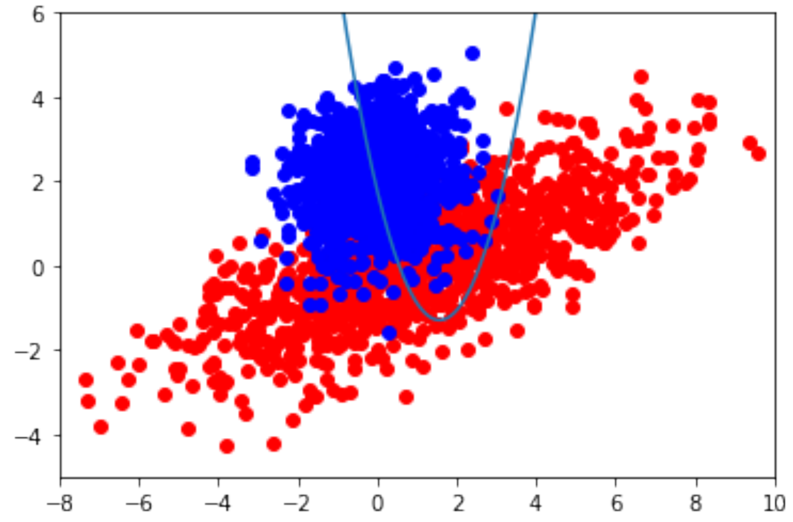
In [2]:

```
M0 = np.array([[1], [0]])
M1 = np.array([[0], [2]])
S0 = np.array([[8, 3], [3, 2]])
S1 = np.array([[1, .1], [.1, 1]])
w = 2 * (np.linalg.inv(S0) @ M0 - np.linalg.inv(S1) @ M1)
c = np.log(np.linalg.det(S0)) - np.log(np.linalg.det(S1))
c = c + M0.T @ np.linalg.inv(S0) @ M0  - M1.T @ np.linalg.inv(S1) @ M1
B = np.linalg.inv(S1) - np.linalg.inv(S0)
```

In [3]:

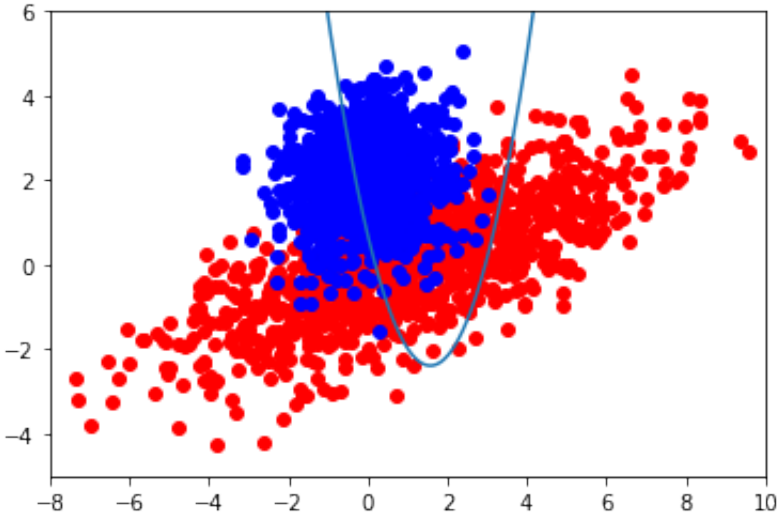
```
# creating 10k instances in each class
X_Y0 = np.random.randn(2, 1000)
A = np.linalg.cholesky(np.array([[8, 3], [3, 2]]))
X_Y0 = A @ X_Y0
for i in range(1000):
    X_Y0[0, i] += 1
    X_Y0[1, i] += 0

X_Y1 = np.random.randn(2, 1000)
A = np.linalg.cholesky(np.array([[1, .1], [.1, 1]]))
X_Y1 = A @ X_Y1
for i in range(1000):
    X_Y1[0, i] += 0
    X_Y1[1, i] += 2

x0 = np.linspace(-20, 20, 10000)
x1 = np.linspace(-20, 20, 10000)
```

In [4]:
```
# plotting decision boundary if uniform prior
boundary = np.zeros([10000,1])
for i in range(x1.shape[0]):
    x = np.array([x0[i], x1[i]])
    boundary[i] = (x.T@B@x + x.T@w - c)


plt.scatter(X_Y0[0, :], X_Y0[1, :], c="r")
plt.scatter(X_Y1[0, :], X_Y1[1, :], c="b")
plt.plot(x0, boundary)
plt.xlim((-8, 10))
plt.ylim((-5, 6))

plt.show()
```

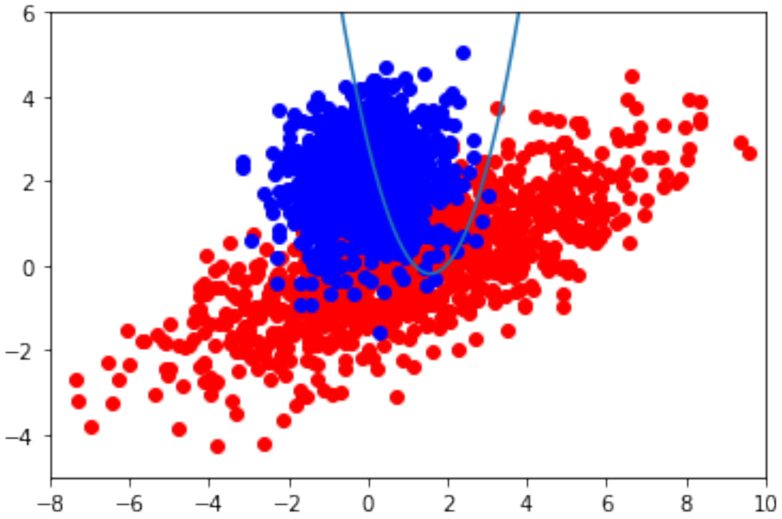


In [5]:

```
# plotting decision boundary if p(y=0) = .25
p_y0 = .25
p_y1 = 1 - p_y0
c_non_uni = c - np.log(p_y0) + np.log(p_y1)
boundary = np.zeros([10000,1])
for i in range(x1.shape[0]):
    x = np.array([x0[i], x1[i]])
    boundary[i] = (x.T@B@x + x.T@w - c_non_uni)


plt.scatter(X_Y0[0, :], X_Y0[1, :], c="r")
plt.scatter(X_Y1[0, :], X_Y1[1, :], c="b")
plt.plot(x0, boundary)
plt.xlim((-8, 10))
plt.ylim((-5, 6))

plt.show()
```



In [6]:
```
# plotting decision boundary if p(y=0) = .75
p_y0 = .75
p_y1 = 1 - p_y0
c_non_uni = c - np.log(p_y0) + np.log(p_y1)
boundary = np.zeros([10000,1])
for i in range(x1.shape[0]):
    x = np.array([x0[i], x1[i]])
    boundary[i] = (x.T@B@x + x.T@w - c_non_uni)


plt.scatter(X_Y0[0, :], X_Y0[1, :], c="r")
plt.scatter(X_Y1[0, :], X_Y1[1, :], c="b")
plt.plot(x0, boundary)
plt.xlim((-8, 10))
plt.ylim((-5, 6))

plt.show()
```



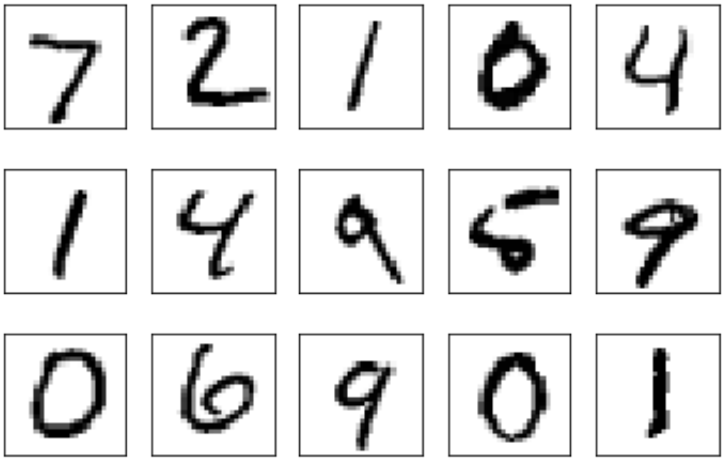
Question 2

a

In [16]:

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_da

## function to plot images in grid
def show_images(images, rows, cols):
    for i in range(rows * cols):
        plt.subplot(rows, cols, i + 1)
        plt.imshow(images[i], cmap=plt.cm.gray_r)
        plt.xticks(())
        plt.yticks(())
    plt.show()

## Uncomment below to see a few images
print('A few example images:')
show_images(x_test, 3, 5)
```

```
A few example images:
```



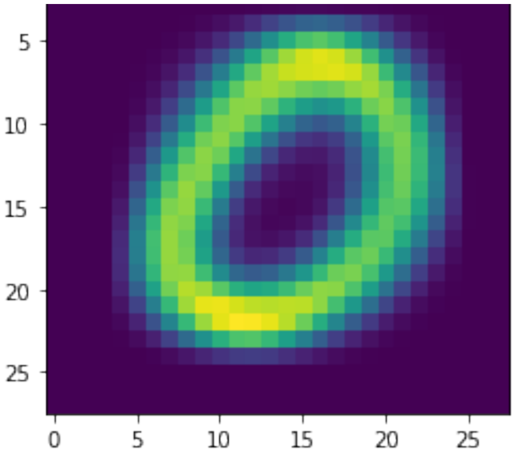
b

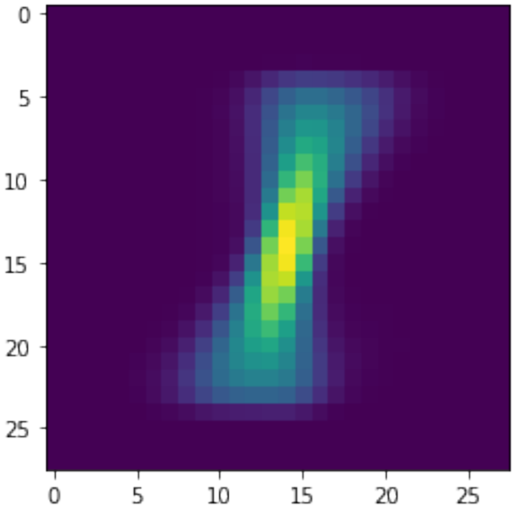
In [8]:

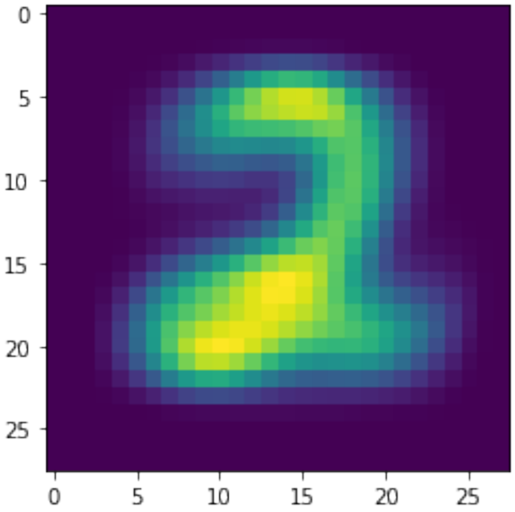
```
mu_ys = np.zeros((784, 10))
sigma_ys = np.zeros((784, 784, 10))

for i in range(10):
    idx = np.argwhere(y_train == i)[:, 0]
    mu_ys[:, i] = (np.mean((x_train[idx]), axis=0)).reshape(-1, 1)[:,
    sigma_ys[:, :, i] = np.cov(x_train[idx].reshape(784, -1))
```

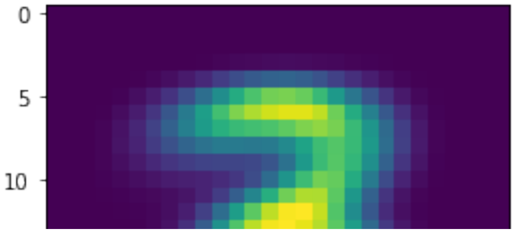
In [9]:

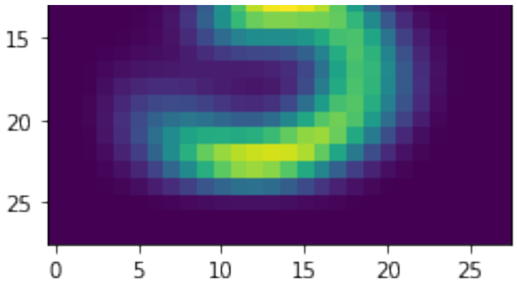
```
for i in range(10):
    plt.figure()
    plt.imshow(mu_ys[:, i].reshape(28, 28))
```

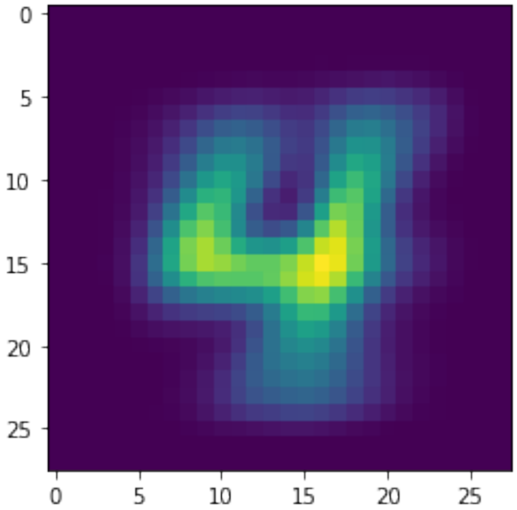

5
10
15
20
25
0
5
10
15
20
25

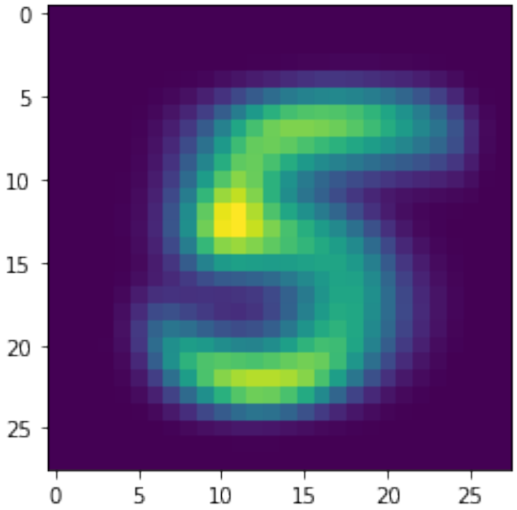

0
5
10
15
20
25
0
5
10
15
20
25

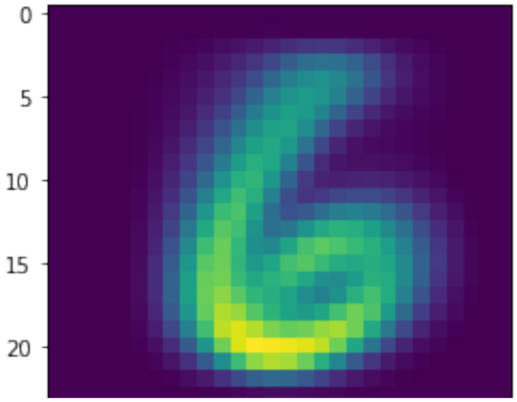

0
5
10
15
20
25
0
5
10
15
20
25

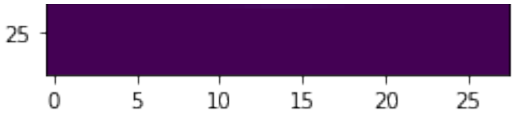

0
5
10

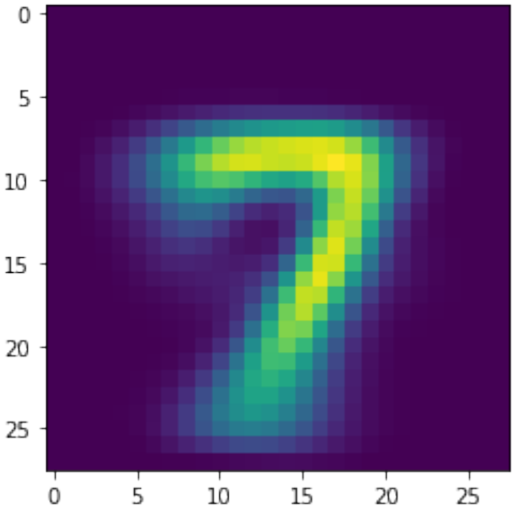

15
20
25
0
5
10
15
20
25

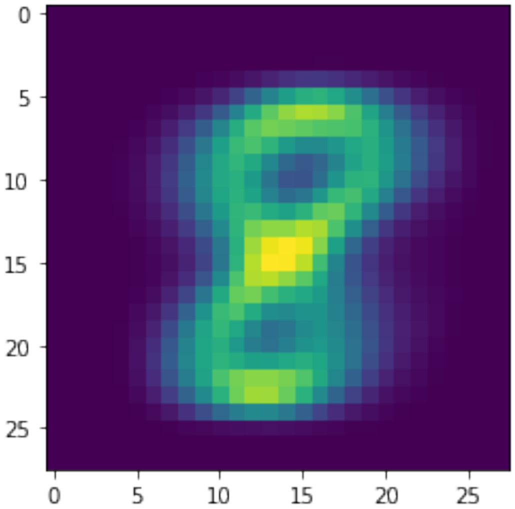

0
5
10
15
20
25
0
5
10
15
20
25

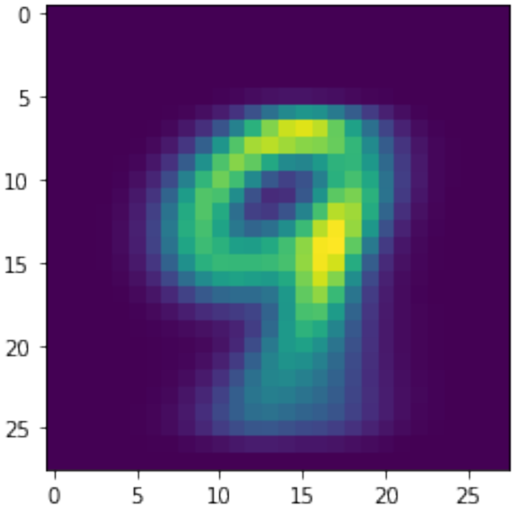

0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20


25
0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25

They look as expected

e

In [36]:

```
def log_likelihood(x, mu, cov_ldet, cov_inv):
    ret =  -1*cov_ldet - (x-mu).T@cov_inv@(x-mu)
    return ret[0, 0]
```

f

In [90]:

```
def prep_params(lam):
    n = 28**2
    mus = np.zeros((784, 10))
    cov_rs = np.zeros((784, 784, 10))
    cov_rinvs = np.zeros((784, 784, 10))
    cov_ldet = np.zeros(10)
    y_heads = np.zeros(len(y_test))

    for i in range(10):
        idx = np.argwhere(y_train == i)[:, 0]
        mus[:, i] = (np.mean((x_train[idx]), axis=0)).reshape(-1, 1)[:
        cov_rs[:, :, i] = np.cov(x_train[idx].reshape(784, -1)) + lam
        cov_rinvs[:, :, i] = np.linalg.inv(cov_rs[:, :, i])
        cov_ldet[i] = np.linalg.slogdet(cov_rs[:, :, i])[1]
```

In [81]:

```
# computing max log likelihood
def run_classsifications():
    for t in range(len(y_test)):
        log_probs = np.zeros(10)
        x = x_test[t].reshape(-1, 1)
        for i in range(10):
            log_probs[i] = log_likelihood(x, mus[:, i].reshape(-1, 1),
        y_heads[t] = np.argmax(log_probs)
```

In [52]:

```
prep_params(1)
run_classsifications()
err_count = np.sum(y_heads != y_test)
print("classification errors = ", err_count/len(y_heads))
```

```
classification errors =  0.3112
```

g

In [67]:
```
ls = np.logspace(-3, 5, 9)
ls = np.delete(ls, 4)
print(ls)
```

```
[1.e-03 1.e-02 1.e-01 1.e+00 1.e+02 1.e+03 1.e+04 1.e+05]
```

In [94]:
```
for lam in ls:
    prep_params(lam)
    run_classsifications()
    err_count = np.sum(y_heads != y_test)
    print("when lamda = {}, classification errors = {}".format(lam, er
```

```
when lamda = 0.001, classification errors = 0.3112
when lamda = 0.01, classification errors = 0.3112
when lamda = 0.1, classification errors = 0.3112
when lamda = 1.0, classification errors = 0.3112
when lamda = 100.0, classification errors = 0.3112
when lamda = 1000.0, classification errors = 0.3112
when lamda = 10000.0, classification errors = 0.3112
when lamda = 100000.0, classification errors = 0.3112
```

g

```
they are all .3112
```