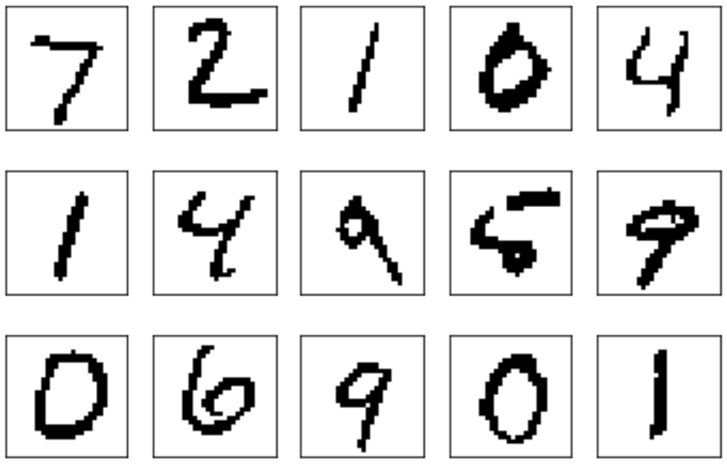
In [2]:
```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from matplotlib import pyplot as plt

(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()

# convert to 0/1 (instead of 0-255)
x_train_int = [np.round(1.0*i/256) for i in x_train]
x_test_int = [np.round(1.0*i/256) for i in x_test]

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

```
A few example images:
```



In [44]:
```
# partition x_train into subsets according to true label
x_parts = [[], [], [], [], [], [], [], [], [], []]
for i in range(60000):
            x_parts[y_train_int[i]] += [i]
```

In [79]:
```
# xj is left pixe, xk is upper pixel
P_MX = np.zeros((10, 2, 4, 4, 28, 28))

for y in range(10):
    for xi_row in range(28):
        for xi_col in range(28):
            for xj in [0, 1]:
                for xk in [0, 1]:
                    prior_count = 0  # count of images that y, xj, xk
                    xi_is_one_count = 0
                    for x_idx in x_parts[y]:
                        cur_img = x_train_int[x_idx]
                        xj_true = xi_col == 0 or cur_img[xi_row][xi_co
                        xk_true = xi_row == 0 or cur_img[xi_row-1][xi_
                        if xj_true and xk_true:
                            prior_count += 1
                            if cur_img[xi_row][xi_col] == 1:
                                xi_is_one_count += 1
                    xi_is_zero_count = prior_count - xi_is_one_count
                    P_MX[y][1][xj][xk][xi_row][xi_col] = (1 + xi_is_or
                    P_MX[y][0][xj][xk][xi_row][xi_col] = (1 + xi_is_ze
```

In [80]:
```
def p_x_given_y_log(x, y):
    p_x_given_y = 1
    for i in range(28):
        for j in range(28):
            xi = x[i][j]
            xj = x[i][j-1] if j != 0 else 1
            xk = x[i-1][j] if i != 0 else 1
#             print((y, xi, xj, xk, i, j))
            xi, xj, xk = int(xi), int(xj), int(xk)
            p_xi = P_MX[y][xi][xj][xk][i][j]
            p_x_given_y *= p_xi
    return np.log(p_x_given_y)

def log_likelihood(x):
    lls = np.zeros((10))
    for y in range(10):
        lls[y] = p_x_given_y_log(x, y)
    return lls
```

In [82]:
```
t_size = len(x_test_int)
err_count = 0
for i in range(t_size):
    y_head = np.argmax(log_likelihood(x_test_int[i]))
    y_label = y_test[i]
    if y_head != y_label:
        err_count += 1
risk = err_count / t_size
print("classification error rate = ", risk)
```

```
classification error rate =  0.0728
```

In [ ]: