In [15]:

```
from keras.datasets import fashion_mnist
(train_X,train_Y), (test_X,test_Y) = fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx
1-ubyte.gz
32768/29515 [=================================] - 0s 3us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx
3-ubyte.gz
26427392/26421880 [==============================] - 9s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-
idx1-ubyte.gz
8192/5148 [===============================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-
idx3-ubyte.gz
4423680/4422102 [==============================] - 1s 0us/step
```

In [16]:

```
import numpy as np
from keras.utils import to_categorical
import matplotlib.pyplot as plt
%matplotlib inline

print('Training data shape : ', train_X.shape, train_Y.shape)

print('Testing data shape : ', test_X.shape, test_Y.shape)
```

```
Training data shape :  (60000, 28, 28) (60000,)
Testing data shape :  (10000, 28, 28) (10000,)
```

In [17]:

```
# Find the unique numbers from the train labels
classes = np.unique(train_Y)
nClasses = len(classes)
print('Total number of outputs : ', nClasses)
print('Output classes : ', classes)
```

```
Total number of outputs :  10
Output classes :  [0 1 2 3 4 5 6 7 8 9]
```

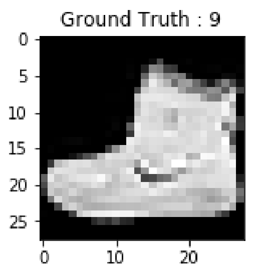
In [18]:

```
plt.figure(figsize=[5,5])

# Display the first image in training data
plt.subplot(121)
plt.imshow(train_X[0,:,:], cmap='gray')
plt.title("Ground Truth : {}".format(train_Y[0]))
```

Out[18]:

```
Text(0.5, 1.0, 'Ground Truth : 9')
```

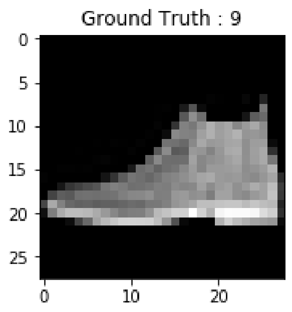


In [19]:

```
# Display the first image in testing data
plt.subplot(122)
plt.imshow(test_X[0,:,:], cmap='gray')
plt.title("Ground Truth : {}".format(test_Y[0]))
```

Out[19]:

```
Text(0.5, 1.0, 'Ground Truth : 9')
```



In [20]:

```
train_X = train_X.reshape(-1, 28,28, 1)
test_X = test_X.reshape(-1, 28,28, 1)
train_X.shape, test_X.shape
```

Out[20]:

```
((60000, 28, 28, 1), (10000, 28, 28, 1))
```

In [21]:

```
train_X = train_X.astype('float32')
test_X = test_X.astype('float32')
train_X = train_X / 255.
test_X = test_X / 255.
```

In [22]:

```
# Change the labels from categorical to one-hot encoding
train_Y_one_hot = to_categorical(train_Y)
test_Y_one_hot = to_categorical(test_Y)

# Display the change for category label using one-hot encoding
print('Original label:', train_Y[0])
print('After conversion to one-hot:', train_Y_one_hot[0])
```

```
Original label: 9
After conversion to one-hot: [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
```

In [23]:

```
from sklearn.model_selection import train_test_split
train_X,valid_X,train_label,valid_label = train_test_split(train_X, train_Y_one_hot, test_size=0.2,
random_state=13)
```

In [24]:

```
train_X.shape,valid_X.shape,train_label.shape,valid_label.shape
```

Out[24]:

```
((48000, 28, 28, 1), (12000, 28, 28, 1), (48000, 10), (12000, 10))
```

In [25]:

```
import keras
from keras.models import Sequential,Input,Model
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras.layers.advanced_activations import LeakyReLU
```

In [26]:

```
batch_size = 64
epochs = 20
num_classes = 10
```

In [28]:

```
fashion_model = Sequential()
fashion_model.add(Conv2D(32, kernel_size=(3, 3),activation='linear',input_shape=(28,28,1),padding='
same'))
fashion_model.add(LeakyReLU(alpha=0.1))
fashion_model.add(MaxPooling2D((2, 2),padding='same'))
fashion_model.add(Conv2D(64, (3, 3), activation='linear',padding='same'))
fashion_model.add(LeakyReLU(alpha=0.1))
fashion_model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
fashion_model.add(Conv2D(128, (3, 3), activation='linear',padding='same'))
fashion_model.add(LeakyReLU(alpha=0.1))
fashion_model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
fashion_model.add(Flatten())
fashion_model.add(Dense(128, activation='linear'))
fashion_model.add(LeakyReLU(alpha=0.1))
fashion_model.add(Dense(num_classes, activation='softmax'))
```

In [29]:

```
fashion_model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adam(
),metrics=['accuracy'])
```

In [30]:

```
fashion_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 32)        320
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 28, 28, 32)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 64)        18496
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 14, 14, 64)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 7, 7, 128)         73856
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 7, 7, 128)         0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 128)               262272
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 356,234
Trainable params: 356,234
```

```
Non-trainable params: 0
_________________________________________________________________
```

In [31]:

```
fashion_train = fashion_model.fit(train_X, train_label, batch_size=batch_size,epochs=epochs,verbose
=1,validation_data=(valid_X, valid_label))
```

```
Epoch 1/20
750/750 [==============================] - 125s 166ms/step - loss: 0.4490 - accuracy: 0.8378 - val
_loss: 0.3198 - val_accuracy: 0.8842
Epoch 2/20
750/750 [==============================] - 118s 157ms/step - loss: 0.2827 - accuracy: 0.8958 - val
_loss: 0.2641 - val_accuracy: 0.9037
Epoch 3/20
750/750 [==============================] - 119s 158ms/step - loss: 0.2430 - accuracy: 0.9104 - val
_loss: 0.2480 - val_accuracy: 0.9090
Epoch 4/20
750/750 [==============================] - 137s 183ms/step - loss: 0.2114 - accuracy: 0.9218 - val
_loss: 0.2368 - val_accuracy: 0.9127
Epoch 5/20
750/750 [==============================] - 135s 181ms/step - loss: 0.1859 - accuracy: 0.9313 - val
_loss: 0.2359 - val_accuracy: 0.9171
Epoch 6/20
750/750 [==============================] - 119s 159ms/step - loss: 0.1653 - accuracy: 0.9387 - val
_loss: 0.2401 - val_accuracy: 0.9152
Epoch 7/20
750/750 [==============================] - 118s 158ms/step - loss: 0.1442 - accuracy: 0.9461 - val
_loss: 0.2412 - val_accuracy: 0.9172
Epoch 8/20
750/750 [==============================] - 114s 153ms/step - loss: 0.1247 - accuracy: 0.9533 - val
_loss: 0.2497 - val_accuracy: 0.9149
Epoch 9/20
750/750 [==============================] - 116s 154ms/step - loss: 0.1048 - accuracy: 0.9596 - val
_loss: 0.2702 - val_accuracy: 0.9155
Epoch 10/20
750/750 [==============================] - 116s 154ms/step - loss: 0.0919 - accuracy: 0.9658 - val
_loss: 0.2819 - val_accuracy: 0.9176
Epoch 11/20
750/750 [==============================] - 120s 159ms/step - loss: 0.0772 - accuracy: 0.9712 - val
_loss: 0.2892 - val_accuracy: 0.9168
Epoch 12/20
750/750 [==============================] - 122s 162ms/step - loss: 0.0683 - accuracy: 0.9743 - val
_loss: 0.3011 - val_accuracy: 0.9192
Epoch 13/20
750/750 [==============================] - 122s 163ms/step - loss: 0.0582 - accuracy: 0.9781 - val
_loss: 0.3257 - val_accuracy: 0.9179
Epoch 14/20
750/750 [==============================] - 123s 164ms/step - loss: 0.0523 - accuracy: 0.9803 - val
_loss: 0.3475 - val_accuracy: 0.9186
Epoch 15/20
750/750 [==============================] - 121s 161ms/step - loss: 0.0490 - accuracy: 0.9820 - val
_loss: 0.3627 - val_accuracy: 0.9197
Epoch 16/20
750/750 [==============================] - 121s 161ms/step - loss: 0.0429 - accuracy: 0.9839 - val
_loss: 0.3877 - val_accuracy: 0.9145
Epoch 17/20
750/750 [==============================] - 121s 161ms/step - loss: 0.0335 - accuracy: 0.9875 - val
_loss: 0.3849 - val_accuracy: 0.9222
Epoch 18/20
750/750 [==============================] - 124s 165ms/step - loss: 0.0361 - accuracy: 0.9865 - val
_loss: 0.4224 - val_accuracy: 0.9182
Epoch 19/20
750/750 [==============================] - 124s 165ms/step - loss: 0.0328 - accuracy: 0.9879 - val
_loss: 0.4464 - val_accuracy: 0.9202
Epoch 20/20
750/750 [==============================] - 129s 172ms/step - loss: 0.0328 - accuracy: 0.9884 - val
_loss: 0.4983 - val_accuracy: 0.9143
```

In [32]:

```
test_eval = fashion_model.evaluate(test_X, test_Y_one_hot, verbose=0)
```

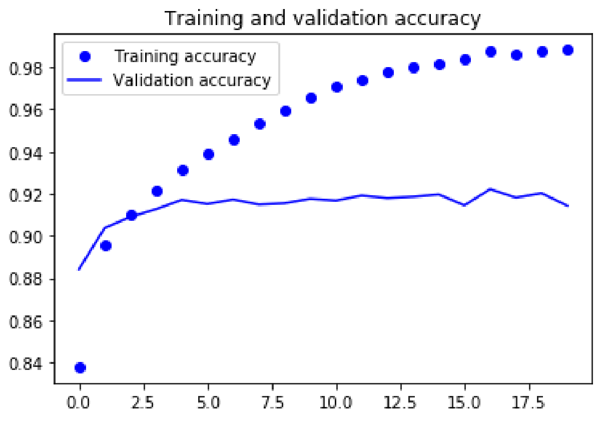
In [33]:

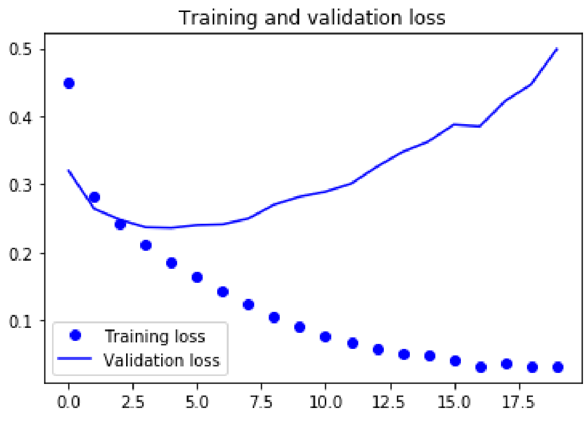
```
print('Test loss:', test_eval[0])
print('Test accuracy:', test_eval[1])
```

```
Test loss: 0.4999641478061676
Test accuracy: 0.913100004196167
```

In [35]:

```
accuracy = fashion_train.history['accuracy']
val_accuracy = fashion_train.history['val_accuracy']
loss = fashion_train.history['loss']
val_loss = fashion_train.history['val_loss']
epochs = range(len(accuracy))
plt.plot(epochs, accuracy, 'bo', label='Training accuracy')
plt.plot(epochs, val_accuracy, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```





In [36]:

```
batch_size = 64
epochs = 20
num_classes = 10
```

In [37]:

```
fashion_model = Sequential()
fashion_model.add(Conv2D(32, kernel_size=(3, 3),activation='linear',padding='same',input_shape=(28,
28,1)))
fashion_model.add(LeakyReLU(alpha=0.1))
fashion_model.add(MaxPooling2D((2, 2),padding='same'))
fashion_model.add(Dropout(0.25))
fashion_model.add(Conv2D(64, (3, 3), activation='linear',padding='same'))
fashion_model.add(LeakyReLU(alpha=0.1))
fashion_model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
fashion_model.add(Dropout(0.25))
fashion_model.add(Conv2D(128, (3, 3), activation='linear',padding='same'))
fashion_model.add(LeakyReLU(alpha=0.1))
fashion_model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
fashion_model.add(Dropout(0.4))
fashion_model.add(Flatten())
fashion_model.add(Dense(128, activation='linear'))
fashion_model.add(LeakyReLU(alpha=0.1))
fashion_model.add(Dropout(0.3))
fashion_model.add(Dense(num_classes, activation='softmax'))
```

In [38]:

```
fashion_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 28, 28, 32)        320
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 28, 28, 32)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 14, 64)        18496
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 14, 14, 64)        0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 7, 7, 128)         73856
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 7, 7, 128)         0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               262272
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 128)               0
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                1290
=================================================================
Total params: 356,234
Trainable params: 356,234
Non-trainable params: 0
_________________________________________________________________
```

In [39]:

```
fashion_model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adam(
),metrics=['accuracy'])
```

In [40]:

```
fashion_train_dropout = fashion_model.fit(train_X, train_label, batch_size=batch_size,epochs=epochs
,verbose=1,validation_data=(valid_X, valid_label))
```

```
Epoch 1/20
```

```
750/750 [==============================] - 122s 162ms/step - loss: 0.5963 - accuracy: 0.7790 - val
_loss: 0.3707 - val_accuracy: 0.8628
Epoch 2/20
750/750 [==============================] - 116s 155ms/step - loss: 0.3777 - accuracy: 0.8614 - val
_loss: 0.3151 - val_accuracy: 0.8852
Epoch 3/20
750/750 [==============================] - 116s 154ms/step - loss: 0.3291 - accuracy: 0.8782 - val
_loss: 0.2823 - val_accuracy: 0.8963
Epoch 4/20
750/750 [==============================] - 116s 155ms/step - loss: 0.3046 - accuracy: 0.8869 - val
_loss: 0.2669 - val_accuracy: 0.8997
Epoch 5/20
750/750 [==============================] - 117s 156ms/step - loss: 0.2830 - accuracy: 0.8960 - val
_loss: 0.2463 - val_accuracy: 0.9092
Epoch 6/20
750/750 [==============================] - 115s 153ms/step - loss: 0.2694 - accuracy: 0.9006 - val
_loss: 0.2433 - val_accuracy: 0.9094
Epoch 7/20
750/750 [==============================] - 116s 155ms/step - loss: 0.2552 - accuracy: 0.9053 - val
_loss: 0.2341 - val_accuracy: 0.9137
Epoch 8/20
750/750 [==============================] - 116s 154ms/step - loss: 0.2517 - accuracy: 0.9061 - val
_loss: 0.2365 - val_accuracy: 0.9153
Epoch 9/20
750/750 [==============================] - 116s 155ms/step - loss: 0.2371 - accuracy: 0.9105 - val
_loss: 0.2305 - val_accuracy: 0.9137
Epoch 10/20
750/750 [==============================] - 117s 156ms/step - loss: 0.2378 - accuracy: 0.9111 - val
_loss: 0.2300 - val_accuracy: 0.9167
Epoch 11/20
750/750 [==============================] - 118s 157ms/step - loss: 0.2288 - accuracy: 0.9146 - val
_loss: 0.2256 - val_accuracy: 0.9172
Epoch 12/20
750/750 [==============================] - 118s 157ms/step - loss: 0.2253 - accuracy: 0.9147 - val
_loss: 0.2201 - val_accuracy: 0.9193
Epoch 13/20
750/750 [==============================] - 120s 160ms/step - loss: 0.2206 - accuracy: 0.9176 - val
_loss: 0.2190 - val_accuracy: 0.9214
Epoch 14/20
750/750 [==============================] - 117s 155ms/step - loss: 0.2146 - accuracy: 0.9195 - val
_loss: 0.2139 - val_accuracy: 0.9231
Epoch 15/20
750/750 [==============================] - 118s 158ms/step - loss: 0.2126 - accuracy: 0.9191 - val
_loss: 0.2190 - val_accuracy: 0.9187
Epoch 16/20
750/750 [==============================] - 118s 158ms/step - loss: 0.2082 - accuracy: 0.9218 - val
_loss: 0.2231 - val_accuracy: 0.9192
Epoch 17/20
750/750 [==============================] - 120s 160ms/step - loss: 0.2078 - accuracy: 0.9214 - val
_loss: 0.2198 - val_accuracy: 0.9214
Epoch 18/20
750/750 [==============================] - 120s 161ms/step - loss: 0.2042 - accuracy: 0.9227 - val
_loss: 0.2176 - val_accuracy: 0.9241
Epoch 19/20
750/750 [==============================] - 117s 156ms/step - loss: 0.2034 - accuracy: 0.9247 - val
_loss: 0.2045 - val_accuracy: 0.9272
Epoch 20/20
750/750 [==============================] - 118s 157ms/step - loss: 0.1978 - accuracy: 0.9252 - val
_loss: 0.2142 - val_accuracy: 0.9262
```

In [41]:

```
fashion_model.save("fashion_model_dropout.h5py")
```

```
WARNING:tensorflow:From C:\Users\DELL\Anaconda3\lib\site-
packages\tensorflow\python\training\tracking\tracking.py:111: Model.state_updates (from
tensorflow.python.keras.engine.training) is deprecated and will be removed in a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatically.
WARNING:tensorflow:From C:\Users\DELL\Anaconda3\lib\site-
packages\tensorflow\python\training\tracking\tracking.py:111: Layer.updates (from
tensorflow.python.keras.engine.base_layer) is deprecated and will be removed in a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatically.
INFO:tensorflow:Assets written to: fashion_model_dropout.h5py\assets
```

In [42]:

```
test_eval = fashion_model.evaluate(test_X, test_Y_one_hot, verbose=1)
```

```
313/313 [==============================] - 7s 22ms/step - loss: 0.2246 - accuracy: 0.9214
```

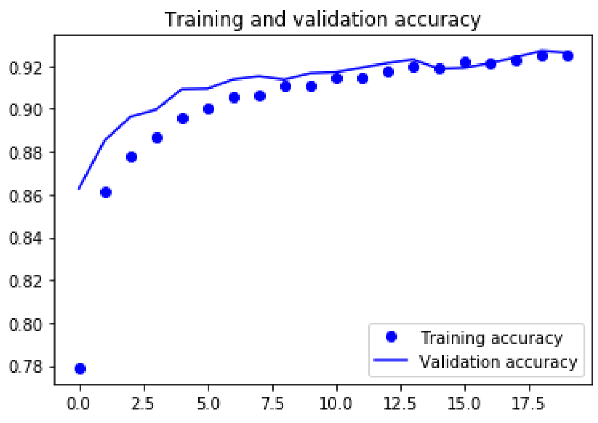
In [43]:

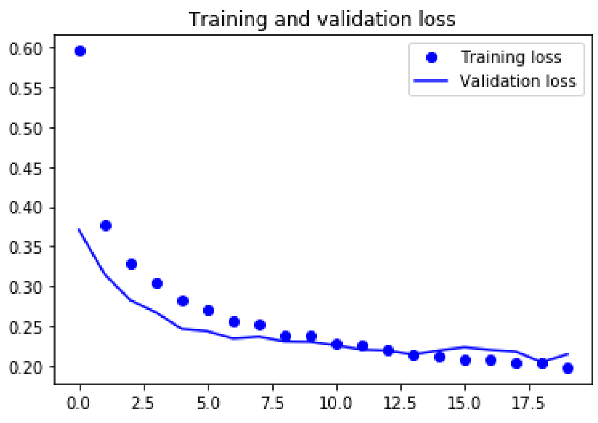
```
print('Test loss:', test_eval[0])
print('Test accuracy:', test_eval[1])
```

```
Test loss: 0.22458960115909576
Test accuracy: 0.9214000105857849
```

In [45]:

```
accuracy = fashion_train_dropout.history['accuracy']
val_accuracy = fashion_train_dropout.history['val_accuracy']
loss = fashion_train_dropout.history['loss']
val_loss = fashion_train_dropout.history['val_loss']
epochs = range(len(accuracy))
plt.plot(epochs, accuracy, 'bo', label='Training accuracy')
plt.plot(epochs, val_accuracy, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```





In [46]:

```
predicted_classes = fashion_model.predict(test_X)
```

In [47]:

```
predicted_classes = np.argmax(np.round(predicted_classes),axis=1)
```

In [48]:

```
predicted_classes.shape, test_Y.shape
```

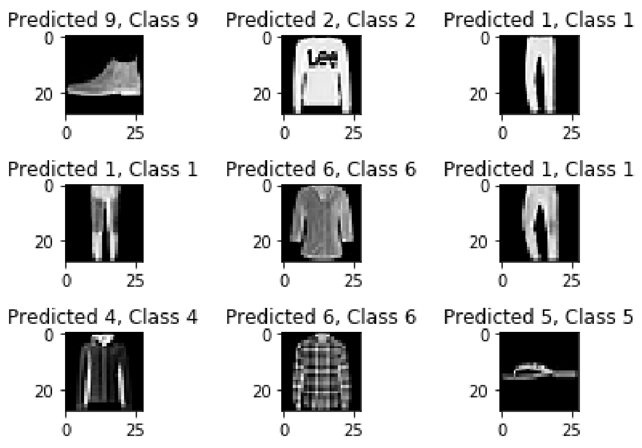
Out[48]:

```
((10000,), (10000,))
```

In [54]:

```
correct = np.where(predicted_classes==test_Y)[0]
print ("Found %d correct labels" % len(correct))
for i, correct in enumerate(correct[:9]):
    plt.subplot(3,3,i+1)
    plt.imshow(test_X[correct].reshape(28,28), cmap='gray', interpolation='none')
    plt.title("Predicted {}, Class {}".format(predicted_classes[correct], test_Y[correct]))
    plt.tight_layout()
```

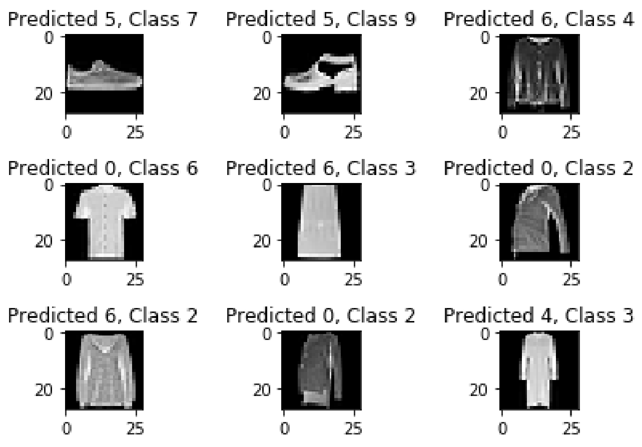
```
Found 9176 correct labels
```



In [53]:

```
incorrect = np.where(predicted_classes!=test_Y)[0]
print ("Found %d incorrect labels" % len(incorrect))
for i, incorrect in enumerate(incorrect[:9]):
    plt.subplot(3,3,i+1)
    plt.imshow(test_X[incorrect].reshape(28,28), cmap='gray', interpolation='none')
    plt.title("Predicted {}, Class {}".format(predicted_classes[incorrect], test_Y[incorrect]))
    plt.tight_layout()
```

```
Found 824 incorrect labels
```



In [55]:

```
from sklearn.metrics import classification_report
target_names = ["Class {}".format(i) for i in range(num_classes)]
print(classification_report(test_Y, predicted_classes, target_names=target_names))
```

```
              precision    recall  f1-score   support

     Class 0       0.80      0.87      0.83      1000
     Class 1       1.00      0.99      0.99      1000
     Class 2       0.91      0.85      0.88      1000
     Class 3       0.93      0.91      0.92      1000
     Class 4       0.87      0.88      0.87      1000
     Class 5       0.99      0.99      0.99      1000
     Class 6       0.77      0.78      0.78      1000
     Class 7       0.96      0.98      0.97      1000
     Class 8       0.99      0.98      0.99      1000
     Class 9       0.98      0.96      0.97      1000

    accuracy                           0.92     10000
   macro avg       0.92      0.92      0.92     10000
weighted avg       0.92      0.92      0.92     10000
```

In [ ]:

```

```