In [2]:

```
# tensorflow와 tf.keras를 임포트
import tensorflow as tf
from tensorflow import keras

from keras.utils import np_utils
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Activation

# 헬퍼(helper) 라이브러리를 임포트
import numpy as np
import matplotlib.pyplot as plt



print(tf.__version__)
```

```
Using TensorFlow backend.
```

```
1.14.0
```

Deep Learning

1. 데이터 셋 설정하기

2. 모델 구성하기

3. 모델 학습과정 설정하기

4.모델 학습시키기

5. 모델 학습과정 살펴보기

6.모델 평가하기

7.모델 사용하기

1. 데이터 셋 설정하기

In [3]:

```
fashion_mnist = keras.datasets.fashion_mnist

(tr_image, tr_label), (te_image, te_label) = fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/tr
ain-labels-idx1-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-dataset
s/train-labels-idx1-ubyte.gz)
32768/29515 [==============================] - 0s 2us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/tr
ain-images-idx3-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-dataset
s/train-images-idx3-ubyte.gz)
26427392/26421880 [==============================] - 5s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t1
0k-labels-idx1-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-dataset
s/t10k-labels-idx1-ubyte.gz)
8192/5148 [===============================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t1
0k-images-idx3-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-dataset
s/t10k-images-idx3-ubyte.gz)
4423680/4422102 [==============================] - 1s 0us/step
```

In [4]:

```
label_name = ['Top', 'Pants', 'Pullover', 'Dress', 'Coat',
              'Sandle', 'Shirt', 'Sneaker', 'Bag', 'Shose']
```

In [5]:

```
tr_image.shape
```

Out[5]:

```
(60000, 28, 28)
```

In [6]:

```
tr_label.shape
```

Out[6]:

```
(60000,)
```

In [7]:

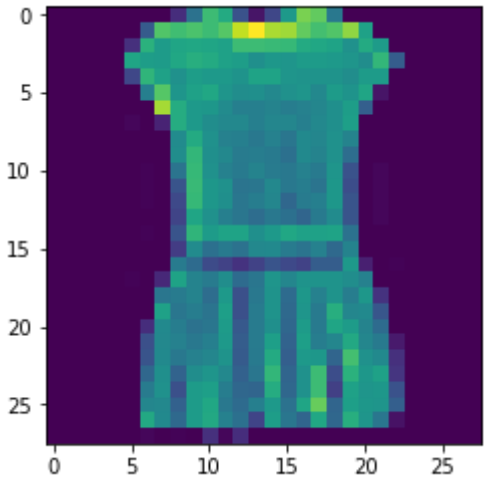
```
tr_label
```

Out[7]:

```
array([9, 0, 0, ..., 3, 0, 5], dtype=uint8)
```

In [11]:

```
plt.figure()
plt.imshow(tr_image[3])

plt.show()
```



In [12]:

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(tr_image[i])
    plt.xlabel(label_name[tr_label[i]])
plt.show()
```

In [0]:

```
tr_image = tr_image.reshape(60000, 784).astype('float32') / 255.0
te_image = te_image.reshape(10000, 784).astype('float32') / 255.0
```

In [0]:

```
tr_label = np_utils.to_categorical(tr_label)
te_label = np_utils.to_categorical(te_label)
```

In [0]:

```
val_image= tr_image[50000:]
val_label= tr_label[50000:]
tr_image = tr_image[:50000]
tr_label = tr_label[:50000]
```

In [16]:

```
val_image.shape
```

Out[16]:

```
(10000, 784)
```

In [0]:

```
tr_rand= np.random.choice(50000, 700)
val_rand = np.random.choice(10000, 300)
```

In [0]:

```
tr_image = tr_image[tr_rand]
tr_label = tr_label[tr_rand]
val_image = val_image[val_rand]
val_label= val_label[val_rand]
```

2. 모델 구성하기

In [19]:

```
model = Sequential()
model.add(Dense(units=64, input_dim=28*28, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(units=10, activation='softmax'))
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorf
low_backend.py:66: The name tf.get_default_graph is deprecated. Please use tf.compa
t.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorf
low_backend.py:541: The name tf.placeholder is deprecated. Please use tf.compat.v1.p
laceholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorf
low_backend.py:4432: The name tf.random_uniform is deprecated. Please use tf.random.
uniform instead.
```

3. 모델 학습과정 설정하기

In [20]:

```
model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:7
93: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimiz
er instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorf
low_backend.py:3576: The name tf.log is deprecated. Please use tf.math.log instead.
```

4. 모델 학습시키기

In [21]:

```
hist = model.fit(tr_image, tr_label, epochs=100, batch_size=32,validation_data=(val_image, val_labe
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/op
s/math_grad.py:1250: add_dispatch_support.<locals>.wrapper (from tensorflow.python.o
ps.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorf
low_backend.py:1033: The name tf.assign_add is deprecated. Please use tf.compat.v1.a
ssign_add instead.

Train on 700 samples, validate on 300 samples
Epoch 1/100
700/700 [==============================] - 1s 1ms/step - loss: 2.2533 - acc: 0.2314
- val_loss: 2.1228 - val_acc: 0.3233
Epoch 2/100
700/700 [==============================] - 0s 63us/step - loss: 2.0138 - acc: 0.4114
- val_loss: 1.9209 - val_acc: 0.4467
Epoch 3/100
700/700 [==============================] - 0s 61us/step - loss: 1.8106 - acc: 0.4857
- val_loss: 1.7322 - val_acc: 0.4867
Epoch 4/100
700/700 [==============================] - 0s 62us/step - loss: 1.6321 - acc: 0.5300
- val_loss: 1.5750 - val_acc: 0.5333
Epoch 5/100
700/700 [==============================] - 0s 63us/step - loss: 1.4742 - acc: 0.5729
- val_loss: 1.4487 - val_acc: 0.5667
Epoch 6/100
700/700 [==============================] - 0s 58us/step - loss: 1.3490 - acc: 0.5957
- val_loss: 1.3314 - val_acc: 0.5967
Epoch 7/100
700/700 [==============================] - 0s 59us/step - loss: 1.2480 - acc: 0.6100
- val_loss: 1.2364 - val_acc: 0.5967
Epoch 8/100
700/700 [==============================] - 0s 63us/step - loss: 1.1646 - acc: 0.6400
- val_loss: 1.1613 - val_acc: 0.6233
Epoch 9/100
700/700 [==============================] - 0s 60us/step - loss: 1.0997 - acc: 0.6343
- val_loss: 1.1032 - val_acc: 0.6467
Epoch 10/100
700/700 [==============================] - 0s 64us/step - loss: 1.0441 - acc: 0.6643
- val_loss: 1.0503 - val_acc: 0.6400
Epoch 11/100
700/700 [==============================] - 0s 60us/step - loss: 0.9970 - acc: 0.6900
- val_loss: 0.9967 - val_acc: 0.6500
Epoch 12/100
700/700 [==============================] - 0s 60us/step - loss: 0.9616 - acc: 0.6786
- val_loss: 0.9724 - val_acc: 0.6700
Epoch 13/100
700/700 [==============================] - 0s 81us/step - loss: 0.9291 - acc: 0.6914
- val_loss: 0.9462 - val_acc: 0.6600
Epoch 14/100
700/700 [==============================] - 0s 65us/step - loss: 0.8981 - acc: 0.6957
- val_loss: 0.9265 - val_acc: 0.6867
Epoch 15/100
700/700 [==============================] - 0s 71us/step - loss: 0.8773 - acc: 0.7043
- val_loss: 0.8913 - val_acc: 0.6800
```

```
Epoch 16/100
700/700 [==============================] - 0s 62us/step - loss: 0.8538 - acc: 0.7057
- val_loss: 0.8766 - val_acc: 0.6867
Epoch 17/100
700/700 [==============================] - 0s 59us/step - loss: 0.8298 - acc: 0.7171
- val_loss: 0.8531 - val_acc: 0.7000
Epoch 18/100
700/700 [==============================] - 0s 61us/step - loss: 0.8123 - acc: 0.7143
- val_loss: 0.8762 - val_acc: 0.6933
Epoch 19/100
700/700 [==============================] - 0s 59us/step - loss: 0.7983 - acc: 0.7300
- val_loss: 0.8323 - val_acc: 0.7100
Epoch 20/100
700/700 [==============================] - 0s 59us/step - loss: 0.7763 - acc: 0.7300
- val_loss: 0.8084 - val_acc: 0.7100
Epoch 21/100
700/700 [==============================] - 0s 63us/step - loss: 0.7642 - acc: 0.7486
- val_loss: 0.7974 - val_acc: 0.7200
Epoch 22/100
700/700 [==============================] - 0s 59us/step - loss: 0.7496 - acc: 0.7557
- val_loss: 0.7964 - val_acc: 0.7333
Epoch 23/100
700/700 [==============================] - 0s 60us/step - loss: 0.7287 - acc: 0.7571
- val_loss: 0.7785 - val_acc: 0.7433
Epoch 24/100
700/700 [==============================] - 0s 61us/step - loss: 0.7171 - acc: 0.7743
- val_loss: 0.7702 - val_acc: 0.7500
Epoch 25/100
700/700 [==============================] - 0s 63us/step - loss: 0.7074 - acc: 0.7671
- val_loss: 0.7555 - val_acc: 0.7500
Epoch 26/100
700/700 [==============================] - 0s 59us/step - loss: 0.6984 - acc: 0.7571
- val_loss: 0.7570 - val_acc: 0.7500
Epoch 27/100
700/700 [==============================] - 0s 60us/step - loss: 0.6827 - acc: 0.7800
- val_loss: 0.7447 - val_acc: 0.7600
Epoch 28/100
700/700 [==============================] - 0s 63us/step - loss: 0.6698 - acc: 0.7814
- val_loss: 0.7359 - val_acc: 0.7433
Epoch 29/100
700/700 [==============================] - 0s 68us/step - loss: 0.6577 - acc: 0.7757
- val_loss: 0.7289 - val_acc: 0.7467
Epoch 30/100
700/700 [==============================] - 0s 67us/step - loss: 0.6472 - acc: 0.7871
- val_loss: 0.7184 - val_acc: 0.7633
Epoch 31/100
700/700 [==============================] - 0s 58us/step - loss: 0.6336 - acc: 0.7900
- val_loss: 0.7142 - val_acc: 0.7600
Epoch 32/100
700/700 [==============================] - 0s 57us/step - loss: 0.6267 - acc: 0.7929
- val_loss: 0.7391 - val_acc: 0.7533
Epoch 33/100
700/700 [==============================] - 0s 61us/step - loss: 0.6112 - acc: 0.7971
- val_loss: 0.7089 - val_acc: 0.7600
Epoch 34/100
700/700 [==============================] - 0s 61us/step - loss: 0.6064 - acc: 0.8129
- val_loss: 0.6871 - val_acc: 0.7567
Epoch 35/100
700/700 [==============================] - 0s 55us/step - loss: 0.5951 - acc: 0.8029
- val_loss: 0.6828 - val_acc: 0.7600
Epoch 36/100
```

```
700/700 [==============================] - 0s 58us/step - loss: 0.5840 - acc: 0.8143
- val_loss: 0.6729 - val_acc: 0.7267
Epoch 37/100
700/700 [==============================] - 0s 60us/step - loss: 0.5779 - acc: 0.8129
- val_loss: 0.6931 - val_acc: 0.7633
Epoch 38/100
700/700 [==============================] - 0s 58us/step - loss: 0.5725 - acc: 0.8171
- val_loss: 0.6711 - val_acc: 0.7533
Epoch 39/100
700/700 [==============================] - 0s 60us/step - loss: 0.5620 - acc: 0.8171
- val_loss: 0.6729 - val_acc: 0.7700
Epoch 40/100
700/700 [==============================] - 0s 59us/step - loss: 0.5524 - acc: 0.8243
- val_loss: 0.6906 - val_acc: 0.7633
Epoch 41/100
700/700 [==============================] - 0s 61us/step - loss: 0.5485 - acc: 0.8186
- val_loss: 0.6806 - val_acc: 0.7667
Epoch 42/100
700/700 [==============================] - 0s 68us/step - loss: 0.5440 - acc: 0.8271
- val_loss: 0.6458 - val_acc: 0.7600
Epoch 43/100
700/700 [==============================] - 0s 63us/step - loss: 0.5344 - acc: 0.8243
- val_loss: 0.6602 - val_acc: 0.7700
Epoch 44/100
700/700 [==============================] - 0s 60us/step - loss: 0.5240 - acc: 0.8214
- val_loss: 0.6611 - val_acc: 0.7400
Epoch 45/100
700/700 [==============================] - 0s 61us/step - loss: 0.5238 - acc: 0.8229
- val_loss: 0.6434 - val_acc: 0.7700
Epoch 46/100
700/700 [==============================] - 0s 61us/step - loss: 0.5131 - acc: 0.8329
- val_loss: 0.6414 - val_acc: 0.7667
Epoch 47/100
700/700 [==============================] - 0s 68us/step - loss: 0.5169 - acc: 0.8100
- val_loss: 0.6662 - val_acc: 0.7867
Epoch 48/100
700/700 [==============================] - 0s 64us/step - loss: 0.5034 - acc: 0.8329
- val_loss: 0.6830 - val_acc: 0.7733
Epoch 49/100
700/700 [==============================] - 0s 59us/step - loss: 0.4901 - acc: 0.8443
- val_loss: 0.6186 - val_acc: 0.7667
Epoch 50/100
700/700 [==============================] - 0s 59us/step - loss: 0.4920 - acc: 0.8243
- val_loss: 0.6263 - val_acc: 0.7633
Epoch 51/100
700/700 [==============================] - 0s 60us/step - loss: 0.4875 - acc: 0.8400
- val_loss: 0.6406 - val_acc: 0.7767
Epoch 52/100
700/700 [==============================] - 0s 63us/step - loss: 0.4768 - acc: 0.8314
- val_loss: 0.6708 - val_acc: 0.7667
Epoch 53/100
700/700 [==============================] - 0s 63us/step - loss: 0.4741 - acc: 0.8514
- val_loss: 0.6216 - val_acc: 0.7533
Epoch 54/100
700/700 [==============================] - 0s 59us/step - loss: 0.4651 - acc: 0.8414
- val_loss: 0.6377 - val_acc: 0.7833
Epoch 55/100
700/700 [==============================] - 0s 62us/step - loss: 0.4594 - acc: 0.8543
- val_loss: 0.6207 - val_acc: 0.7467
Epoch 56/100
700/700 [==============================] - 0s 63us/step - loss: 0.4630 - acc: 0.8343
```

```
- val_loss: 0.6234 - val_acc: 0.7767
Epoch 57/100
700/700 [==============================] - 0s 62us/step - loss: 0.4516 - acc: 0.8529
- val_loss: 0.6088 - val_acc: 0.7767
Epoch 58/100
700/700 [==============================] - 0s 66us/step - loss: 0.4426 - acc: 0.8500
- val_loss: 0.6696 - val_acc: 0.7700
Epoch 59/100
700/700 [==============================] - 0s 64us/step - loss: 0.4420 - acc: 0.8443
- val_loss: 0.6087 - val_acc: 0.7700
Epoch 60/100
700/700 [==============================] - 0s 74us/step - loss: 0.4315 - acc: 0.8600
- val_loss: 0.6318 - val_acc: 0.7833
Epoch 61/100
700/700 [==============================] - 0s 71us/step - loss: 0.4253 - acc: 0.8586
- val_loss: 0.6033 - val_acc: 0.7500
Epoch 62/100
700/700 [==============================] - 0s 65us/step - loss: 0.4281 - acc: 0.8571
- val_loss: 0.6658 - val_acc: 0.7700
Epoch 63/100
700/700 [==============================] - 0s 61us/step - loss: 0.4212 - acc: 0.8657
- val_loss: 0.6155 - val_acc: 0.7700
Epoch 64/100
700/700 [==============================] - 0s 63us/step - loss: 0.4255 - acc: 0.8643
- val_loss: 0.5997 - val_acc: 0.7867
Epoch 65/100
700/700 [==============================] - 0s 60us/step - loss: 0.4124 - acc: 0.8671
- val_loss: 0.6248 - val_acc: 0.7733
Epoch 66/100
700/700 [==============================] - 0s 57us/step - loss: 0.4066 - acc: 0.8657
- val_loss: 0.6171 - val_acc: 0.7800
Epoch 67/100
700/700 [==============================] - 0s 60us/step - loss: 0.4005 - acc: 0.8629
- val_loss: 0.6487 - val_acc: 0.7800
Epoch 68/100
700/700 [==============================] - 0s 63us/step - loss: 0.3968 - acc: 0.8714
- val_loss: 0.6189 - val_acc: 0.7767
Epoch 69/100
700/700 [==============================] - 0s 58us/step - loss: 0.3992 - acc: 0.8729
- val_loss: 0.5854 - val_acc: 0.7733
Epoch 70/100
700/700 [==============================] - 0s 62us/step - loss: 0.3880 - acc: 0.8743
- val_loss: 0.5981 - val_acc: 0.7800
Epoch 71/100
700/700 [==============================] - 0s 60us/step - loss: 0.3832 - acc: 0.8657
- val_loss: 0.6009 - val_acc: 0.7567
Epoch 72/100
700/700 [==============================] - 0s 59us/step - loss: 0.3772 - acc: 0.8843
- val_loss: 0.6013 - val_acc: 0.7733
Epoch 73/100
700/700 [==============================] - 0s 61us/step - loss: 0.3722 - acc: 0.8843
- val_loss: 0.5852 - val_acc: 0.7767
Epoch 74/100
700/700 [==============================] - 0s 62us/step - loss: 0.3765 - acc: 0.8743
- val_loss: 0.5865 - val_acc: 0.7867
Epoch 75/100
700/700 [==============================] - 0s 70us/step - loss: 0.3743 - acc: 0.8700
- val_loss: 0.6062 - val_acc: 0.7733
Epoch 76/100
700/700 [==============================] - 0s 65us/step - loss: 0.3652 - acc: 0.8929
- val_loss: 0.6073 - val_acc: 0.7900
```

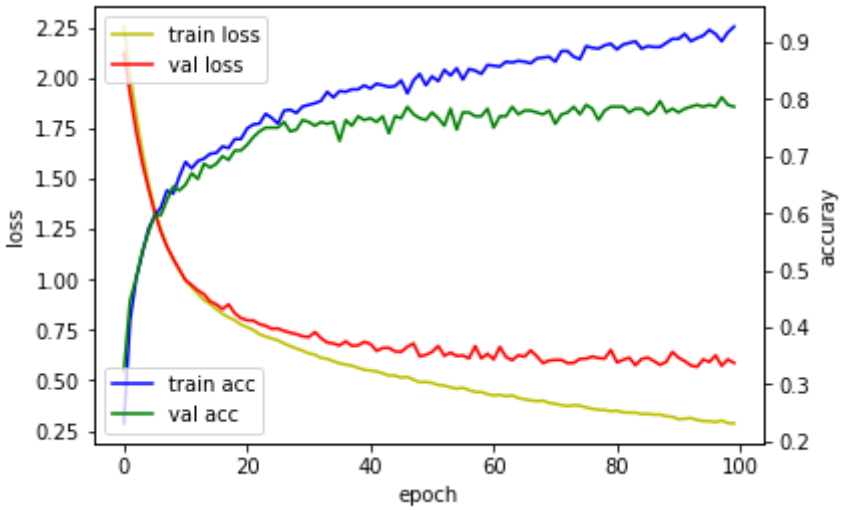
```
Epoch 77/100
700/700 [==============================] - 0s 62us/step - loss: 0.3588 - acc: 0.8900
- val_loss: 0.6177 - val_acc: 0.7833
Epoch 78/100
700/700 [==============================] - 0s 60us/step - loss: 0.3544 - acc: 0.8886
- val_loss: 0.6108 - val_acc: 0.7633
Epoch 79/100
700/700 [==============================] - 0s 59us/step - loss: 0.3527 - acc: 0.8943
- val_loss: 0.6072 - val_acc: 0.7800
Epoch 80/100
700/700 [==============================] - 0s 63us/step - loss: 0.3467 - acc: 0.8971
- val_loss: 0.6512 - val_acc: 0.7867
Epoch 81/100
700/700 [==============================] - 0s 60us/step - loss: 0.3497 - acc: 0.8871
- val_loss: 0.5889 - val_acc: 0.7867
Epoch 82/100
700/700 [==============================] - 0s 62us/step - loss: 0.3411 - acc: 0.8957
- val_loss: 0.6014 - val_acc: 0.7867
Epoch 83/100
700/700 [==============================] - 0s 60us/step - loss: 0.3400 - acc: 0.8986
- val_loss: 0.6125 - val_acc: 0.7767
Epoch 84/100
700/700 [==============================] - 0s 61us/step - loss: 0.3398 - acc: 0.9014
- val_loss: 0.5747 - val_acc: 0.7833
Epoch 85/100
700/700 [==============================] - 0s 59us/step - loss: 0.3327 - acc: 0.8886
- val_loss: 0.6062 - val_acc: 0.7833
Epoch 86/100
700/700 [==============================] - 0s 60us/step - loss: 0.3335 - acc: 0.8929
- val_loss: 0.6129 - val_acc: 0.7733
Epoch 87/100
700/700 [==============================] - 0s 60us/step - loss: 0.3298 - acc: 0.8914
- val_loss: 0.5959 - val_acc: 0.7800
Epoch 88/100
700/700 [==============================] - 0s 58us/step - loss: 0.3304 - acc: 0.8914
- val_loss: 0.5763 - val_acc: 0.7967
Epoch 89/100
700/700 [==============================] - 0s 60us/step - loss: 0.3222 - acc: 0.8986
- val_loss: 0.5921 - val_acc: 0.7767
Epoch 90/100
700/700 [==============================] - 0s 62us/step - loss: 0.3192 - acc: 0.9057
- val_loss: 0.6408 - val_acc: 0.7833
Epoch 91/100
700/700 [==============================] - 0s 64us/step - loss: 0.3078 - acc: 0.9057
- val_loss: 0.6119 - val_acc: 0.7767
Epoch 92/100
700/700 [==============================] - 0s 62us/step - loss: 0.3095 - acc: 0.9143
- val_loss: 0.5955 - val_acc: 0.7833
Epoch 93/100
700/700 [==============================] - 0s 59us/step - loss: 0.3130 - acc: 0.9014
- val_loss: 0.5736 - val_acc: 0.7867
Epoch 94/100
700/700 [==============================] - 0s 60us/step - loss: 0.3032 - acc: 0.9071
- val_loss: 0.5669 - val_acc: 0.7900
Epoch 95/100
700/700 [==============================] - 0s 58us/step - loss: 0.2983 - acc: 0.9114
- val_loss: 0.6035 - val_acc: 0.7867
Epoch 96/100
700/700 [==============================] - 0s 66us/step - loss: 0.2979 - acc: 0.9214
- val_loss: 0.5912 - val_acc: 0.7900
Epoch 97/100
```

```
700/700 [==============================] - 0s 60us/step - loss: 0.2931 - acc: 0.9143
- val_loss: 0.6239 - val_acc: 0.7867
Epoch 98/100
700/700 [==============================] - 0s 58us/step - loss: 0.3006 - acc: 0.9014
- val_loss: 0.5718 - val_acc: 0.8033
Epoch 99/100
700/700 [==============================] - 0s 59us/step - loss: 0.2882 - acc: 0.9171
- val_loss: 0.6039 - val_acc: 0.7900
Epoch 100/100
700/700 [==============================] - 0s 59us/step - loss: 0.2861 - acc: 0.9271
- val_loss: 0.5861 - val_acc: 0.7867
```

5. 모델 학습과정 살펴보기

In [22]:

```
%matplotlib inline
import matplotlib.pyplot as plt

fig, loss_ax = plt.subplots()

acc_ax = loss_ax.twinx()

loss_ax.plot(hist.history['loss'], 'y', label='train loss')
loss_ax.plot(hist.history['val_loss'], 'r', label='val loss')

acc_ax.plot(hist.history['acc'], 'b', label='train acc')
acc_ax.plot(hist.history['val_acc'], 'g', label='val acc')

loss_ax.set_xlabel('epoch')
loss_ax.set_ylabel('loss')
acc_ax.set_ylabel('accuray')

loss_ax.legend(loc='upper left')
acc_ax.legend(loc='lower left')

plt.show()
```



In [23]:

```
print('## training loss and acc ##')
print(hist.history['loss'])
print(hist.history['acc'])
```

```
## training loss and acc ##
[2.253254019873483, 2.01377781527383, 1.81064498765128, 1.6320535101209368, 1.474175
387791225, 1.3490218905040197, 1.2479833105632236, 1.1645606136322022, 1.09965991497
0398, 1.0441101448876517, 0.9969519676480975, 0.961566264969962, 0.9290939508165632,
0.8981269254003252, 0.8773463528496879, 0.8537975403240748, 0.8298169704845973, 0.81
23071554728917, 0.7982828232220242, 0.776327576977866, 0.7641930273600988, 0.7496366
064889091, 0.728667128426688, 0.7170600380216327, 0.7073664821897234, 0.698381745474
6791, 0.6827476242610386, 0.6698061639922006, 0.6577159377506802, 0.647213560853685
6, 0.6336285287993295, 0.6266872828347342, 0.6112055626937322, 0.6064174951825823,
0.595068780694689, 0.584038564477648, 0.5779397685187203, 0.5724510373388018, 0.5620
249683516366, 0.5523693067686898, 0.5484671854972839, 0.5440388679504394, 0.53440238
91857692, 0.5239505508967809, 0.5237863899980273, 0.5130883032935006, 0.516900452205
113, 0.5033530970982143, 0.49014548063278196, 0.4920216883931841, 0.487495647498539
5, 0.47681489774159025, 0.47409529532705036, 0.4651281746796199, 0.4593760582378932
4, 0.463007378748485, 0.4516040749209268, 0.4425635984965733, 0.44202457002231055,
0.4314988994598389, 0.4252570038182395, 0.4280942402567182, 0.42124195541654313, 0.4
2552580169269016, 0.4123536023071834, 0.40656061359814233, 0.40045505591801234, 0.39
68419410501208, 0.39921573434557234, 0.3879708470617022, 0.3831766424860273, 0.37716
55467578343, 0.3721636491162436, 0.37654913783073424, 0.3742801415920258, 0.36520185
74305943, 0.35884419049535476, 0.35441138352666585, 0.3527343898160117, 0.3467197418
2128906, 0.34969882130622865, 0.34110245398112704, 0.33998803070613315, 0.3397744590
895517, 0.33274315748895916, 0.33350240639277867, 0.3297838262149266, 0.330400895220
8928, 0.32215886286326817, 0.3191830291918346, 0.30775334596633913, 0.30948005276066
914, 0.31304850714547294, 0.30318463512829374, 0.29826700823647634, 0.29787092174802
51, 0.2930545676606042, 0.3005953824520111, 0.28819286329405647, 0.2861289041382925
6]
[0.23142857142857143, 0.4114285709176745, 0.48571428537368777, 0.530000000340598, 0.
5728571438789367, 0.5957142860548836, 0.61, 0.6400000006811959, 0.6342857139451163,
0.6642857132639204, 0.6900000010217939, 0.6785714278902326, 0.6914285717691694, 0.69
57142850330897, 0.7042857132639204, 0.7057142853736877, 0.7171428574834551, 0.714285
7153075082, 0.730000001021794, 0.730000001021794, 0.7485714289120265, 0.755714286054
8837, 0.7571428561210632, 0.7742857149669102, 0.7671428578240531, 0.757142858164651
1, 0.779999999659402, 0.7814285724503653, 0.7757142867360797, 0.787142858164651, 0.7
89999999318804, 0.7928571428571428, 0.797142858164651, 0.8128571428571428, 0.8028571
42175947, 0.8142857136045183, 0.8128571431977408, 0.8171428568022592, 0.817142856121
0632, 0.8242857149669103, 0.8185714295932225, 0.8271428564616612, 0.824285713604518
3, 0.8214285717691694, 0.8228571438789367, 0.8328571428571429, 0.8099999993188041,
0.832857141835349, 0.8442857142857143, 0.8242857149669103, 0.84, 0.8314285724503654,
0.8514285717691694, 0.8414285710879734, 0.8542857146263123, 0.8342857153075082, 0.85
2857141835349, 0.850000000340598, 0.8442857132639203, 0.860000000340598, 0.858571427
5496347, 0.8571428568022592, 0.8657142853736878, 0.8642857136045183, 0.8671428561210
632, 0.8657142853736878, 0.8628571435383388, 0.8714285717691694, 0.8728571425165449,
0.8742857139451163, 0.8657142860548837, 0.8842857146263122, 0.8842857153075082, 0.87
42857139451163, 0.870000000340598, 0.8928571431977408, 0.8899999989782061, 0.8885714
295932224, 0.8942857149669102, 0.8971428568022591, 0.8871428574834551, 0.89571428469
24918, 0.8985714282308306, 0.9014285710879735, 0.8885714292526246, 0.892857142516544
9, 0.8914285717691695, 0.8914285704067775, 0.8985714282308306, 0.9057142846924918,
0.9057142857142857, 0.9142857142857143, 0.9014285704067775, 0.9071428568022591, 0.91
14285704067775, 0.9214285724503654, 0.9142857149669102, 0.9014285724503653, 0.917142
8568022592, 0.927142857824053]
```

6. 모델 평가하기

In [24]:

```
loss_and_acc = model.evaluate(te_image, te_label, batch_size=32)
print('## evaluation  ##')
print(loss_and_acc)
```

```
10000/10000 [==============================] - 0s 22us/step
## evaluation  ##
[0.6234635022640228, 0.7841]
```

7. 모델 사용하기

In [25]:

```
xhat = te_image
yhat = model.predict(xhat)
print('## yhat ##')
print(yhat)
```

```
## yhat ##
[[1.22700085e-05 3.10547284e-06 1.59673209e-06 ... 4.69718784e-01
  2.18283315e-03 3.91369611e-01]
 [1.25898208e-04 2.99526200e-05 8.71882677e-01 ... 7.38897921e-09
  6.33818272e-05 2.24308341e-08]
 [4.34300091e-06 9.99864340e-01 2.11641673e-05 ... 2.78694151e-10
  4.70053187e-08 1.04688125e-09]
 ...
 [6.26689428e-03 1.57258255e-05 7.96275272e-04 ... 2.40807130e-04
  9.07908320e-01 8.30771096e-05]
 [4.60590818e-05 9.90254104e-01 3.07097594e-04 ... 5.09890242e-07
  8.69520409e-06 1.36579990e-06]
 [3.25186411e-04 7.01888988e-04 3.30585288e-04 ... 3.43248546e-01
  1.56451687e-02 3.39865163e-02]]
```

In [26]:

```
te_image.shape
```

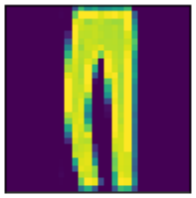
Out[26]:

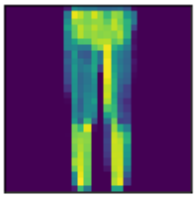
```
(10000, 784)
```

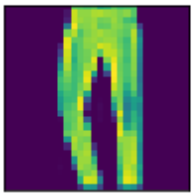
In [29]:

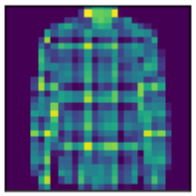
```
%matplotlib inline
import matplotlib.pyplot as plt



#plt_row = 10
#plt_col = 10


i = 0


plt.figure(figsize=(10,10))
for i in range(25):
    if np.argmax(te_label[i]) == np.argmax(yhat[i]):
        plt.subplot(5,5,i+1)
        plt.xticks([])
        plt.yticks([])
        plt.grid(False)
        plt.imshow(te_image[i].reshape(28, 28))
        plt.xlabel('R: ' + str(np.argmax(te_label[i])) + ' P: ' + str(np.argmax(yhat[i])))
        i += 1


plt.show()
```

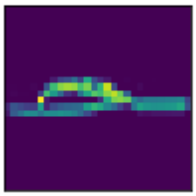

R: 2 P: 2

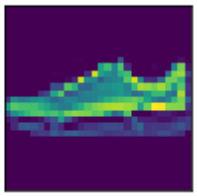

R: 1 P: 1

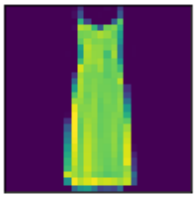

R: 1 P: 1

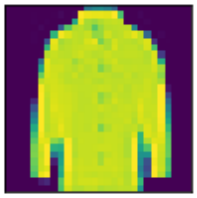

R: 1 P: 1

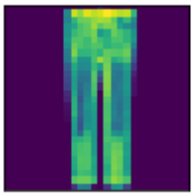

R: 6 P: 6

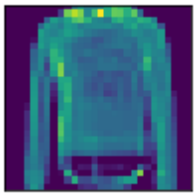

R: 5 P: 5

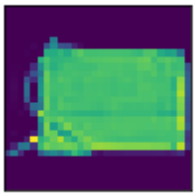

R: 7 P: 7


R: 3 P: 3

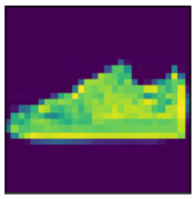

R: 4 P: 4

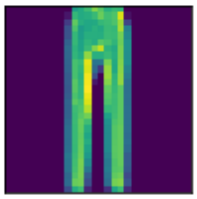

R: 1 P: 1


R: 2 P: 2


R: 8 P: 8

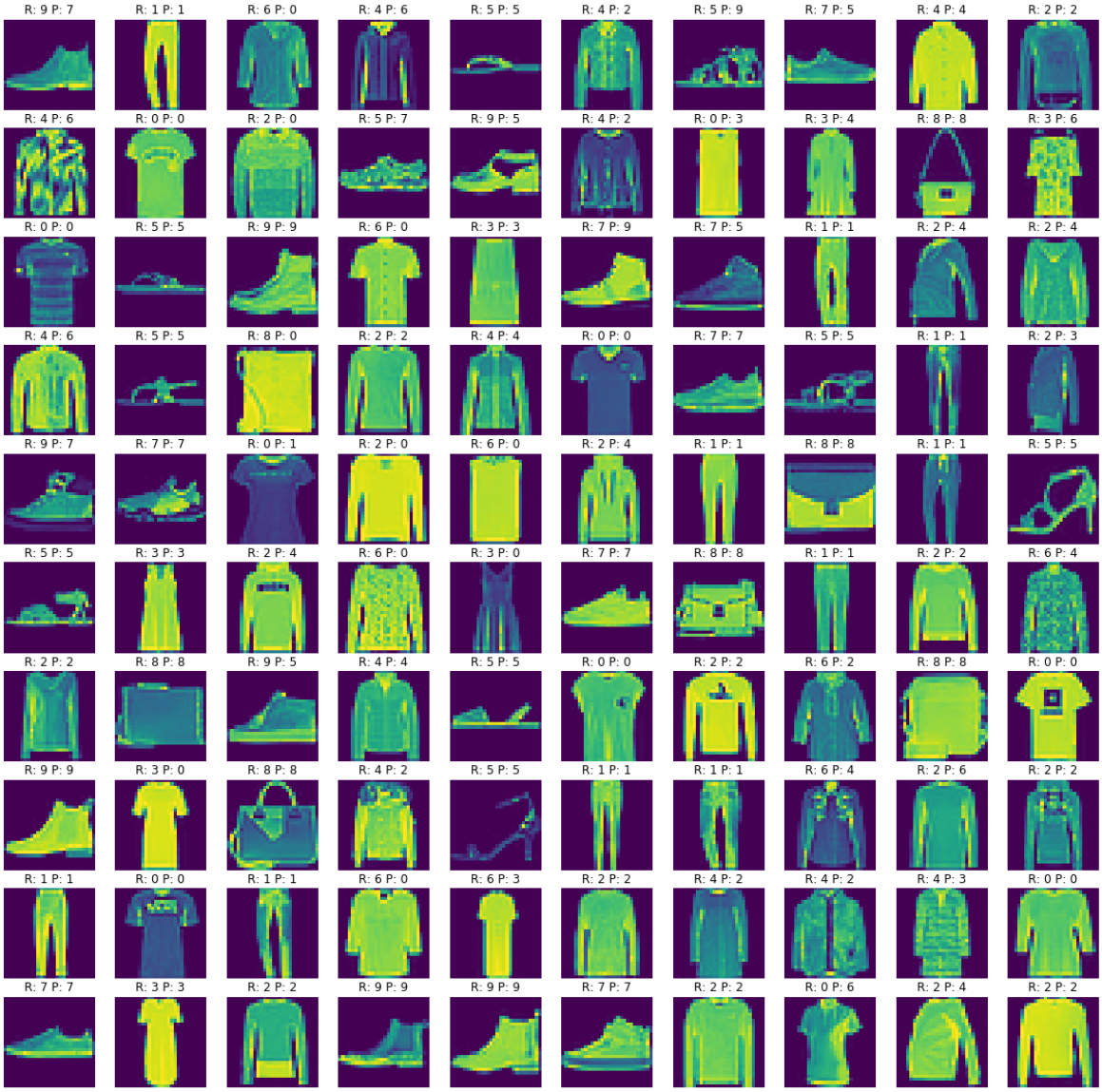

R: 0 P: 0


R: 7 P: 7


R: 1 P: 1

In [28]:

```
%matplotlib inline
import matplotlib.pyplot as plt



plt_row = 10
plt_col = 10

plt.rcParams["figure.figsize"] = (20,20)

f, axarr = plt.subplots(plt_row, plt_col)

cnt = 0
i = 0

while cnt < (plt_row*plt_col):

    if np.argmax(te_label[i]) == np.argmax(yhat[i]):
        i += 1
      # continue

    sub_plt=axarr[cnt//plt_row, cnt%plt_col]
    sub_plt.axis('off')
    sub_plt.imshow(te_image[i].reshape(28, 28))
    sub_plt_title = 'R: ' + str(np.argmax(te_label[i])) + ' P: ' + str(np.argmax(yhat[i]))
    sub_plt.set_title(sub_plt_title)

    i += 1
    cnt += 1

plt.show()
```


R: 9 P: 7
R: 1 P: 1
R: 6 P: 0
R: 4 P: 6
R: 5 P: 5
R: 4 P: 2
R: 5 P: 9
R: 7 P: 5
R: 4 P: 4
R: 2 P: 2
R: 4 P: 6
R: 0 P: 0
R: 2 P: 0
R: 5 P: 7
R: 9 P: 5
R: 4 P: 2
R: 0 P: 3
R: 3 P: 4
R: 8 P: 8
R: 3 P: 6
R: 0 P: 0
R: 5 P: 5
R: 9 P: 9
R: 6 P: 0
R: 3 P: 3
R: 7 P: 9
R: 7 P: 5
R: 1 P: 1
R: 2 P: 4
R: 2 P: 4
R: 4 P: 6
R: 5 P: 5
R: 8 P: 0
R: 2 P: 2
R: 4 P: 4
R: 0 P: 0
R: 7 P: 7
R: 5 P: 5
R: 1 P: 1
R: 2 P: 3
R: 9 P: 7
R: 7 P: 7
R: 0 P: 1
R: 2 P: 0
R: 6 P: 0
R: 2 P: 4
R: 1 P: 1
R: 8 P: 8
R: 1 P: 1
R: 5 P: 5
R: 5 P: 5
R: 3 P: 3
R: 2 P: 4
R: 6 P: 0
R: 3 P: 0
R: 7 P: 7
R: 8 P: 8
R: 1 P: 1
R: 2 P: 2
R: 6 P: 4
R: 2 P: 2
R: 8 P: 8
R: 9 P: 5
R: 4 P: 4
R: 5 P: 5
R: 0 P: 0
R: 2 P: 2
R: 6 P: 2
R: 8 P: 8
R: 0 P: 0
R: 9 P: 9
R: 3 P: 0
R: 8 P: 8
R: 4 P: 2
R: 5 P: 5
R: 1 P: 1
R: 1 P: 1
R: 6 P: 4
R: 2 P: 6
R: 2 P: 2
R: 1 P: 1
R: 0 P: 0
R: 1 P: 1
R: 6 P: 0
R: 6 P: 3
R: 2 P: 2
R: 4 P: 2
R: 4 P: 2
R: 4 P: 3
R: 0 P: 0
R: 7 P: 7
R: 3 P: 3
R: 2 P: 2
R: 9 P: 9
R: 9 P: 9
R: 7 P: 7
R: 2 P: 2
R: 0 P: 6
R: 2 P: 4
R: 2 P: 2