In [2]:

```
# Do the imports, so that we write less code

from sklearn.cross_validation import train_test_split
import numpy as np
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.optimizers import SGD, Adam, RMSprop
from keras.utils import np_utils
from keras import backend as K
from keras.callbacks import EarlyStopping

import matplotlib.pyplot as plt
import time
from datetime import datetime
from PIL import Image, ImageDraw

# from keras.utils.visualize_util import plot
# from IPython.display import SVG
# from keras.utils.visualize_util import model_to_dot
```

```
Using Theano backend.

Couldn't import dot_parser, loading of dot files will not be p
ossible.
```

In [ ]:

```
# Prepare and load the data for analysis
import scipy.io
from sklearn.decomposition import PCA
mat = scipy.io.loadmat('2013MT60597.mat')
raw_data = mat['data_image']
target = mat['data_labels']
target = target.flatten()

pca = PCA(n_components=0.9)
transformed_data = pca.fit_transform(raw_data)


X_train, X_test, Y_train, y_test = train_test_split(transformed_data, targ
et, test_size=0.10, random_state=42)

n_samples, n_features = X_train.shape
n_classes = len(np.unique(target))

Y_train = np_utils.to_categorical(Y_train, n_classes)
Y_test = np_utils.to_categorical(y_test, n_classes)
```

In [82]:

```
# getImage(X_train[0])
```

Out[82]:

In [3]:

```
#define the function for getting images
def getImage(x):
    img =  Image.fromarray(255 - (x.reshape(28, 28)).astype('uint8'))
#     plt.imshow(img,cmap='Greys_r')
#     plt.show()
    return img
```

In [83]:

```
# SVG(model_to_dot(model).create(prog='dot', format='svg'))
```

In [7]:

```
epoch = 10
train = []
test = []
time = []
early_stopping = EarlyStopping(monitor='val_loss', patience=5, verbose=1,
mode='auto')
neurons = [25,50,100,150,200,300,400,500,600,700,800,1000]
# lr = [0.0000001,0.000001,0.00001,0.0001,0.001,0.01,0.1,1,10]
for i in range(len(neurons)):

    model = Sequential()
    #Add the first layer
    model.add(Dense(neurons[i],input_dim=784))
    model.add(Activation('sigmoid'))
    #Add the second layer
    model.add(Dense(10))
    model.add(Activation("sigmoid"))
    #model.summary()


    model.compile(loss='categorical_crossentropy',
                  optimizer=RMSprop(),
                  metrics=['accuracy'])
    model.optimizer.lr.set_value(lr[i])

    print("Training new model")
    a = datetime.now()
    history = model.fit(X_train, Y_train,nb_epoch=epoch,verbose=0, callbac
k=[early_stopping] validation_data=(X_test, Y_test))
    b = datetime.now()
    time.append((b-a).total_seconds())
    train.append(history.history.get("acc")[-1])
    test.append(history.history.get("val_acc")[-1])
    print("Iteration finished")
```

```
Training new model
Iteration finished
Training new model
Iteration finished
Training new model
Iteration finished
Training new model
Iteration finished
Training new model
Iteration finished
Training new model
Iteration finished
Training new model
Iteration finished
Training new model
Iteration finished
Training new model
Iteration finished
Training new model
Iteration finished
Training new model
Iteration finished
```

In [120]:

```
# predicted = model.predict_classes(X_test)
```

```
200/200 [==============================] - 0s
```

In [128]:

```
# for i in range(200):
#     if predicted[i] != y_test[i]:
#         img = getImage(X_test[i])
# #         a = "misclassified/"
#         a = str(predicted[i])
#         a += "Iter"
#         a += str(i)
#         a += ".bmp"
#         img.save(a)
```

In [9]:

```
plt.figure()#figsize=(8, 6)

plt.title('Accuracy vs hidden neurons')

#plt.figure(figsize=(8, 6))
#plt.subplots_adjust(left=0.05, right=0.95, bottom=0.15, top=0.95)
#plt.imshow(accuracy, interpolation='nearest', cmap=plt.cm.spectral)
plt.xlabel('Learning Rate')
plt.ylabel('Accuracy ( in %)')
plt.plot(train,'g-')
plt.plot(test,'r-')
plt.legend(['train accuracy','test accuracy'], loc='upper left')
#plt.ylabel('some numbers')
#plt.title('Cross validated Test error -' + kernel1)
#plt.colorbar()
#plt.yticks(np.arange(len(gamma1)), gamma1, rotation=45)
plt.xticks(np.arange(len(neurons)), neurons)
#plt.text(C1.index(max_index),1,max_acc)
plt.savefig('mlp.png')
plt.show()
```

In [11]:

```
plt.figure()#figsize=(8, 6)

plt.title('Time vs hidden neurons')

#plt.figure(figsize=(8, 6))
#plt.subplots_adjust(left=0.05, right=0.95, bottom=0.15, top=0.95)
#plt.imshow(accuracy, interpolation='nearest', cmap=plt.cm.spectral)
plt.xlabel('Learning rate')
plt.ylabel('Time (in sec)')
plt.plot(time)
#plt.ylabel('some numbers')
#plt.title('Cross validated Test error -' + kernel1)
#plt.colorbar()
#plt.yticks(np.arange(len(gamma1)), gamma1, rotation=45)
plt.xticks(np.arange(len(neurons)), neurons)
#plt.text(C1.index(max_index),1,max_acc)
plt.savefig('mlp_time.png')
plt.show()
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```