In [1]:

```
import tensorflow as tf
import numpy as np
import pickle as pkl
from sklearn.manifold import TSNE
from flip_gradient import flip_gradient
from utils import *
from timeit import default_timer as timer
import numpy as np
from keras import backend as K
from keras.datasets import mnist
from keras.layers import Conv2D,Dense,MaxPool2D,Flatten
from keras.models import Sequential
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from keras_helper import NNWeightHelper
from snes import SNES
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import scipy.misc
```

```
/anaconda2/lib/python2.7/site-packages/h5py/__init__.py:36: FutureWarning: Conversi
on of the second argument of issubdtype from `float` to `np.floating` is deprecate
d. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

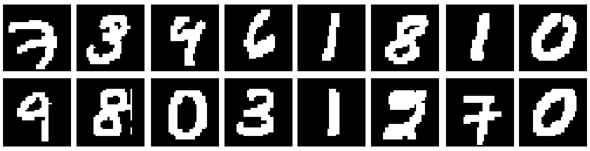
In [2]:

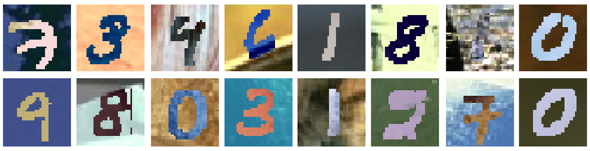
```

from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets('MNIST_data', one_hot=True)

# Process MNIST
mnist_train = (mnist.train.images > 0).reshape(55000, 28, 28, 1).astype(np.uint8) * 25
mnist_train = np.concatenate([mnist_train, mnist_train, mnist_train], 3)
mnist_test = (mnist.test.images > 0).reshape(10000, 28, 28, 1).astype(np.uint8) * 255
mnist_test = np.concatenate([mnist_test, mnist_test, mnist_test], 3)
mnist_valid = (mnist.validation.images > 0).reshape(mnist.validation.images.shape[0],2
mnist_valid= np.concatenate([mnist_valid,mnist_valid,mnist_valid],axis=3)


print(type(mnist_train[0]))

# Load MNIST-M
mnistm = pkl.load(open('mnistm_data.pkl', 'rb'))
mnistm_train = mnistm['train']
mnistm_test = mnistm['test']
mnistm_valid = mnistm['valid']

# Compute pixel mean for normalizing data
pixel_mean = np.vstack([mnist_train, mnistm_train]).mean((0, 1, 2))

# Create a mixed dataset for TSNE visualization
num_test = 500
combined_test_imgs = np.vstack([mnist_test[:num_test], mnistm_test[:num_test]])
combined_test_labels = np.vstack([mnist.test.labels[:num_test], mnist.test.labels[:num
combined_test_domain = np.vstack([np.tile([1., 0.], [num_test, 1]),
        np.tile([0., 1.], [num_test, 1])])

```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
<type 'numpy.ndarray'>
```

In [3]:

```
imshow_grid(mnist_train)
imshow_grid(mnistm_train)
```





In [4]:

```
mnist_train_labels = mnist.train.labels
mnist_test_labels = mnist.test.labels
mnist_valid_labels = mnist.validation.labels
print (len(mnist_train_labels), len(mnist_test_labels))
```

```
(55000, 10000)
```

In [5]:

```
# input image dimensions
img_rows, img_cols = 28, 28
num_classes = 10
input_shape = (img_rows, img_cols, 3)
```

In [6]:

```
x_train, y_train = mnist_train , mnist_train_labels
x_valid, y_valid = mnist_valid, mnist_valid_labels
Mx_train, My_train = mnistm_train , mnist_train_labels
Mx_valid, My_valid = mnistm_valid, mnist_valid_labels
```

In [7]:

```
model = Sequential()
model.add(Conv2D(32,kernel_size=(5,5),activation='relu',input_shape=input_shape))
model.add(MaxPool2D())
model.add((Conv2D(48,kernel_size=(3, 3), activation='relu')))
model.add(MaxPool2D())
model.add(Flatten())
model.add(Dense(20,activation='relu'))
model.compile(loss='mse',optimizer='adam')
model = model
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 24, 24, 32)        2432      
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 32)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 10, 48)        13872     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 48)          0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 1200)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 20)                24020     
=================================================================
Total params: 40,324
Trainable params: 40,324
Non-trainable params: 0
_________________________________________________________________
```

In [8]:

```
nnw = NNWeightHelper(model)
weights = nnw.get_weights()
```

In [11]:

```
SX=np.asarray(x_train)
SY=np.asarray(y_train)
SVX=np.asarray(x_valid)
SVY=np.asarray(y_valid)
input_shape=SX.shape[1:]

TX = np.asarray(Mx_train)
TY = np.asarray(My_train)
TVX = np.asarray(Mx_valid)
TVY = np.asarray(My_valid)

MX=np.concatenate((SX, TX), axis=0)
MY=np.concatenate((np.zeros(SX.shape[0]), np.ones(TX.shape[0])),axis=0)

MVX = np.concatenate((SVX, TVX),axis=0)
MVY = np.concatenate((np.zeros(SVX.shape[0]), np.ones(TVX.shape[0])),axis=0)

input_shape=SX.shape[1:]
```

In [12]:

```
def Train_classifier(x,y):
    x_features=model.predict(x)
    clf=RandomForestClassifier(n_estimators=18)
    clf=clf.fit(x_features, y)
    return clf

def Predict_from_clf(clf,x):
    x_features=model.predict(x)
    y=clf.predict(x_features)
    return y

def saveModel(filename):
    file=open(filename+'.json','w')
    file.write(model.to_json())
    model.save_weights(filename+'.h5')
    file.close()

def load_model(filename):
    file=open(filename+'.json')
    model=file.read()
    model=model_from_json(model)
    model.load_weight(filename+'.json')

def new_score_fun(label_accuracy,domain_accuray):
    temp =label_accuracy-domain_accuray
    temp /= max(domain_accuray, label_accuracy)
    return temp
```

In [13]:

```
label_classifier = RandomForestClassifier()
Domain_classifier = RandomForestClassifier()
```

In [14]:

```
RD_Indices = np.random.choice(a=list(range(MX.shape[0])),size=1024)
M_D_TX = MX[RD_Indices]
M_D_TY = MY[RD_Indices]

RS_Indices = np.random.choice(a=list(range(SX.shape[0])),size=4024)
S_TX = SX[RS_Indices]
S_TY = SY[RS_Indices]

RDV_indices = np.random.choice(a=list(range(MVX.shape[0])), size=1024)
M_D_VX = MVX[RDV_indices]
M_D_VY = MVY[RDV_indices]

SV_indices = np.random.choice(a=list((range(SVX.shape[0]))),size=1024)
validX = SVX[SV_indices]
validY = SVY[SV_indices]
```

In [15]:

```
label_classifier = Train_classifier(SX,SY)
label_pred = Predict_from_clf(label_classifier, validX)
label_accuracy = accuracy_score(validY,label_pred)
print ('label_accuracy',label_accuracy )

Domain_classifier = Train_classifier(M_D_TX, M_D_TY)
domain_pred = Predict_from_clf(Domain_classifier, M_D_VX)
domain_accuracy = accuracy_score(M_D_VY, domain_pred)
print ('domain_accuracy',domain_accuracy )

weight_modifier=NNWeightHelper(model)
weights=weight_modifier.get_weights()

print 'weights to evolve:',len(weights)
```

```
('label_accuracy', 0.3662109375)
('domain_accuracy', 0.8291015625)
weights to evolve: 40324
```

In [16]:

```
snes=SNES(weights,1,20)
```

In [17]:

```
for i in range(60):
    start = timer()
    new_weights=snes.ask()

    complied_score = []
    domain_accuracys = []
    label_accuracys = []

    for w in new_weights:
        weight_modifier.set_weights(w)

        label_classifier=Train_classifier(S_TX,S_TY)
        label_predictions = Predict_from_clf(label_classifier, validX)
        label_accuracy=accuracy_score(validY,label_predictions)
#         print ('label_accuracy',label_accuracy )

        Domain_classifier = Train_classifier(M_D_TX,M_D_TY)
        domain_predictions=Predict_from_clf(Domain_classifier,M_D_VX)
        domain_accuracy=accuracy_score(M_D_VY,domain_predictions)
#         print ('domain_accuracy',domain_accuracy )


        complied_score.append(new_score_fun(label_accuracy, domain_accuracy))
        domain_accuracys.append(domain_accuracy)
        label_accuracys.append(label_accuracy)

    snes.tell(new_weights,complied_score)
    most_fit_model=np.argmax(complied_score)
    end = timer()

    print("It took", end - start, "seconds to complete generation")
    print("the fit model has label_accuracy: %0.3f and domain_accuracy:%0.3f and fitne
          %(label_accuracys[most_fit_model], domain_accuracys[most_fit_model], complie
```

```
('Step', 1, ':', -0.5269978401727862, 'best:', -0.5269978401727862, 20)
('It took', 40.6933331489563, 'seconds to complete generation')
the fit model has label_accuracy: 0.428 and domain_accuracy:0.904 and fitness_scor
e:-0.527
('Step', 2, ':', -0.4723404255319149, 'best:', -0.4723404255319149, 20)
('It took', 40.73397397994995, 'seconds to complete generation')
the fit model has label_accuracy: 0.484 and domain_accuracy:0.918 and fitness_scor
e:-0.472
('Step', 3, ':', -0.4851063829787234, 'best:', -0.4723404255319149, 20)
('It took', 36.06898593902588, 'seconds to complete generation')
the fit model has label_accuracy: 0.473 and domain_accuracy:0.918 and fitness_scor
e:-0.485
('Step', 4, ':', -0.4630593132154006, 'best:', -0.4630593132154006, 20)
('It took', 36.09901690483093, 'seconds to complete generation')
the fit model has label_accuracy: 0.504 and domain_accuracy:0.938 and fitness_scor
e:-0.463
('Step', 5, ':', -0.44081632653061226, 'best:', -0.44081632653061226, 20)
('It took', 35.73264694213867, 'seconds to complete generation')
the fit model has label_accuracy: 0.535 and domain_accuracy:0.957 and fitness_scor
e:-0.441
('Step', 6, ':', -0.4629237288135593, 'best:', -0.44081632653061226, 20)
('It took', 38.45079207420349, 'seconds to complete generation')
the fit model has label_accuracy: 0.495 and domain_accuracy:0.922 and fitness_scor
e:-0.463
('Step', 7, ':', -0.4647739221871714, 'best:', -0.44081632653061226, 20)
('It took', 38.4210479259491, 'seconds to complete generation')
the fit model has label_accuracy: 0.497 and domain_accuracy:0.929 and fitness_scor
e:-0.465
('Step', 8, ':', -0.5048025613660619, 'best:', -0.44081632653061226, 20)
('It took', 38.471384048461914, 'seconds to complete generation')
the fit model has label_accuracy: 0.453 and domain_accuracy:0.915 and fitness_scor
e:-0.505
('Step', 9, ':', -0.4309031556039173, 'best:', -0.4309031556039173, 20)
('It took', 38.18576383590698, 'seconds to complete generation')
```

```
the fit model has label_accuracy: 0.511 and domain_accuracy:0.897 and fitness_scor
e:-0.431
('Step', 10, ':', -0.39397089397089397, 'best:', -0.39397089397089397, 20)
('It took', 36.95579695701599, 'seconds to complete generation')
the fit model has label_accuracy: 0.569 and domain_accuracy:0.939 and fitness_scor
e:-0.394
('Step', 11, ':', -0.4166666666666667, 'best:', -0.39397089397089397, 20)
('It took', 39.99688196182251, 'seconds to complete generation')
the fit model has label_accuracy: 0.540 and domain_accuracy:0.926 and fitness_scor
e:-0.417
('Step', 12, ':', -0.3935617860851506, 'best:', -0.3935617860851506, 20)
('It took', 37.453953981399536, 'seconds to complete generation')
the fit model has label_accuracy: 0.570 and domain_accuracy:0.940 and fitness_scor
e:-0.394
('Step', 13, ':', -0.37733887733887733, 'best:', -0.37733887733887733, 20)
('It took', 39.587193965911865, 'seconds to complete generation')
the fit model has label_accuracy: 0.585 and domain_accuracy:0.939 and fitness_scor
e:-0.377
('Step', 14, ':', -0.37563971340839303, 'best:', -0.37563971340839303, 20)
('It took', 39.667535066604614, 'seconds to complete generation')
the fit model has label_accuracy: 0.596 and domain_accuracy:0.954 and fitness_scor
e:-0.376
('Step', 15, ':', -0.3302083333333333, 'best:', -0.3302083333333333, 20)
('It took', 38.27355885505676, 'seconds to complete generation')
the fit model has label_accuracy: 0.628 and domain_accuracy:0.938 and fitness_scor
e:-0.330
('Step', 16, ':', -0.32659574468085106, 'best:', -0.32659574468085106, 20)
('It took', 40.71008801460266, 'seconds to complete generation')
the fit model has label_accuracy: 0.618 and domain_accuracy:0.918 and fitness_scor
e:-0.327
('Step', 17, ':', -0.30062630480167013, 'best:', -0.30062630480167013, 20)
('It took', 40.482568979263306, 'seconds to complete generation')
the fit model has label_accuracy: 0.654 and domain_accuracy:0.936 and fitness_scor
e:-0.301
('Step', 18, ':', -0.3316008316008316, 'best:', -0.30062630480167013, 20)
('It took', 41.086037158966064, 'seconds to complete generation')
the fit model has label_accuracy: 0.628 and domain_accuracy:0.939 and fitness_scor
e:-0.332
('Step', 19, ':', -0.33127572016460904, 'best:', -0.30062630480167013, 20)
('It took', 41.590157985687256, 'seconds to complete generation')
the fit model has label_accuracy: 0.635 and domain_accuracy:0.949 and fitness_scor
e:-0.331
('Step', 20, ':', -0.335423197492163, 'best:', -0.30062630480167013, 20)
('It took', 40.2183141708374, 'seconds to complete generation')
the fit model has label_accuracy: 0.621 and domain_accuracy:0.935 and fitness_scor
e:-0.335
('Step', 21, ':', -0.3266171792152704, 'best:', -0.30062630480167013, 20)
('It took', 39.08803200721741, 'seconds to complete generation')
the fit model has label_accuracy: 0.620 and domain_accuracy:0.921 and fitness_scor
e:-0.327
('Step', 22, ':', -0.3050847457627119, 'best:', -0.30062630480167013, 20)
('It took', 40.74550986289978, 'seconds to complete generation')
the fit model has label_accuracy: 0.641 and domain_accuracy:0.922 and fitness_scor
e:-0.305
('Step', 23, ':', -0.2861602497398543, 'best:', -0.2861602497398543, 20)
('It took', 40.2285680770874, 'seconds to complete generation')
the fit model has label_accuracy: 0.670 and domain_accuracy:0.938 and fitness_scor
e:-0.286
('Step', 24, ':', -0.3044838373305527, 'best:', -0.2861602497398543, 20)
('It took', 39.85899114608765, 'seconds to complete generation')
the fit model has label_accuracy: 0.651 and domain_accuracy:0.937 and fitness_scor
e:-0.304
('Step', 25, ':', -0.29958677685950413, 'best:', -0.2861602497398543, 20)
('It took', 38.89574599266052, 'seconds to complete generation')
the fit model has label_accuracy: 0.662 and domain_accuracy:0.945 and fitness_scor
e:-0.300
('Step', 26, ':', -0.2806282722513089, 'best:', -0.2806282722513089, 20)
('It took', 41.240391969680786, 'seconds to complete generation')
the fit model has label_accuracy: 0.671 and domain_accuracy:0.933 and fitness_scor
e:-0.281
```

```
('Step', 27, ':', -0.2864637985309549, 'best:', -0.2806282722513089, 20)
('It took', 38.86373496055603, 'seconds to complete generation')
the fit model has label_accuracy: 0.664 and domain_accuracy:0.931 and fitness_scor
e:-0.286
('Step', 28, ':', -0.2843551797040169, 'best:', -0.2806282722513089, 20)
('It took', 42.67580008506775, 'seconds to complete generation')
the fit model has label_accuracy: 0.661 and domain_accuracy:0.924 and fitness_scor
e:-0.284
('Step', 29, ':', -0.2753468516542156, 'best:', -0.2753468516542156, 20)
('It took', 40.232882022857666, 'seconds to complete generation')
the fit model has label_accuracy: 0.663 and domain_accuracy:0.915 and fitness_scor
e:-0.275
('Step', 30, ':', -0.273784355179704, 'best:', -0.273784355179704, 20)
('It took', 42.03857088088989, 'seconds to complete generation')
the fit model has label_accuracy: 0.671 and domain_accuracy:0.924 and fitness_scor
e:-0.274
('Step', 31, ':', -0.27415966386554624, 'best:', -0.273784355179704, 20)
('It took', 40.271496057510376, 'seconds to complete generation')
the fit model has label_accuracy: 0.675 and domain_accuracy:0.930 and fitness_scor
e:-0.274
('Step', 32, ':', -0.26834381551362685, 'best:', -0.26834381551362685, 20)
('It took', 41.6530499458313, 'seconds to complete generation')
the fit model has label_accuracy: 0.682 and domain_accuracy:0.932 and fitness_scor
e:-0.268
('Step', 33, ':', -0.24444444444444444, 'best:', -0.24444444444444444, 20)
('It took', 38.141144037246704, 'seconds to complete generation')
the fit model has label_accuracy: 0.697 and domain_accuracy:0.923 and fitness_scor
e:-0.244
('Step', 34, ':', -0.25937834941050375, 'best:', -0.24444444444444444, 20)
('It took', 43.60002088546753, 'seconds to complete generation')
the fit model has label_accuracy: 0.675 and domain_accuracy:0.911 and fitness_scor
e:-0.259
('Step', 35, ':', -0.2708113804004215, 'best:', -0.24444444444444444, 20)
('It took', 41.69706583023071, 'seconds to complete generation')
the fit model has label_accuracy: 0.676 and domain_accuracy:0.927 and fitness_scor
e:-0.271
('Step', 36, ':', -0.2611464968152866, 'best:', -0.24444444444444444, 20)
('It took', 40.41336393356323, 'seconds to complete generation')
the fit model has label_accuracy: 0.680 and domain_accuracy:0.920 and fitness_scor
e:-0.261

('Step', 37, ':', -0.2572972972972973, 'best:', -0.24444444444444444, 20)
('It took', 41.0063841342926, 'seconds to complete generation')
the fit model has label_accuracy: 0.671 and domain_accuracy:0.903 and fitness_scor
e:-0.257
('Step', 38, ':', -0.2604166666666667, 'best:', -0.24444444444444444, 20)
('It took', 42.19031596183777, 'seconds to complete generation')
the fit model has label_accuracy: 0.693 and domain_accuracy:0.938 and fitness_scor
e:-0.260
('Step', 39, ':', -0.20613107822410148, 'best:', -0.20613107822410148, 20)
('It took', 41.03705310821533, 'seconds to complete generation')
the fit model has label_accuracy: 0.733 and domain_accuracy:0.924 and fitness_scor
e:-0.206
('Step', 40, ':', -0.2601880877742947, 'best:', -0.20613107822410148, 20)
('It took', 43.284093141555786, 'seconds to complete generation')
the fit model has label_accuracy: 0.691 and domain_accuracy:0.935 and fitness_scor
e:-0.260
('Step', 41, ':', -0.24572649572649571, 'best:', -0.20613107822410148, 20)
('It took', 39.87771987915039, 'seconds to complete generation')
the fit model has label_accuracy: 0.689 and domain_accuracy:0.914 and fitness_scor
e:-0.246
('Step', 42, ':', -0.2603613177470776, 'best:', -0.20613107822410148, 20)
('It took', 41.78398895263672, 'seconds to complete generation')
the fit model has label_accuracy: 0.680 and domain_accuracy:0.919 and fitness_scor
e:-0.260
('Step', 43, ':', -0.20979765708200213, 'best:', -0.20613107822410148, 20)
('It took', 39.65928912162781, 'seconds to complete generation')
the fit model has label_accuracy: 0.725 and domain_accuracy:0.917 and fitness_scor
e:-0.210
('Step', 44, ':', -0.23435843054082714, 'best:', -0.20613107822410148, 20)
```

```
('It took', 39.42806696891785, 'seconds to complete generation')
the fit model has label_accuracy: 0.705 and domain_accuracy:0.921 and fitness_scor
e:-0.234
('Step', 45, ':', -0.21264994547437296, 'best:', -0.20613107822410148, 20)
('It took', 41.2503879070282, 'seconds to complete generation')
the fit model has label_accuracy: 0.705 and domain_accuracy:0.896 and fitness_scor
e:-0.213
('Step', 46, ':', -0.208067940552017, 'best:', -0.20613107822410148, 20)
('It took', 40.72658705711365, 'seconds to complete generation')
the fit model has label_accuracy: 0.729 and domain_accuracy:0.920 and fitness_scor
e:-0.208
('Step', 47, ':', -0.20276008492569003, 'best:', -0.20276008492569003, 20)
('It took', 39.3925940990448, 'seconds to complete generation')
the fit model has label_accuracy: 0.733 and domain_accuracy:0.920 and fitness_scor
e:-0.203
('Step', 48, ':', -0.20191285866099895, 'best:', -0.20191285866099895, 20)
('It took', 39.929206132888794, 'seconds to complete generation')
the fit model has label_accuracy: 0.733 and domain_accuracy:0.919 and fitness_scor
e:-0.202
('Step', 49, ':', -0.20234291799787008, 'best:', -0.20191285866099895, 20)
('It took', 41.22449803352356, 'seconds to complete generation')
the fit model has label_accuracy: 0.731 and domain_accuracy:0.917 and fitness_scor
e:-0.202
('Step', 50, ':', -0.17282608695652174, 'best:', -0.17282608695652174, 20)
('It took', 40.8322970867157, 'seconds to complete generation')
the fit model has label_accuracy: 0.743 and domain_accuracy:0.898 and fitness_scor
e:-0.173
('Step', 51, ':', -0.18763326226012794, 'best:', -0.17282608695652174, 20)
('It took', 40.65230393409729, 'seconds to complete generation')
the fit model has label_accuracy: 0.744 and domain_accuracy:0.916 and fitness_scor
e:-0.188
('Step', 52, ':', -0.1819148936170213, 'best:', -0.17282608695652174, 20)
('It took', 39.593459129333496, 'seconds to complete generation')
the fit model has label_accuracy: 0.751 and domain_accuracy:0.918 and fitness_scor
e:-0.182
('Step', 53, ':', -0.19894736842105262, 'best:', -0.17282608695652174, 20)
('It took', 41.122143030166626, 'seconds to complete generation')
the fit model has label_accuracy: 0.743 and domain_accuracy:0.928 and fitness_scor
e:-0.199
('Step', 54, ':', -0.19135135135135134, 'best:', -0.17282608695652174, 20)
('It took', 43.967583894729614, 'seconds to complete generation')
the fit model has label_accuracy: 0.730 and domain_accuracy:0.903 and fitness_scor
e:-0.191
('Step', 55, ':', -0.18619246861924685, 'best:', -0.17282608695652174, 20)
('It took', 40.919296979904175, 'seconds to complete generation')
the fit model has label_accuracy: 0.760 and domain_accuracy:0.934 and fitness_scor
e:-0.186
('Step', 56, ':', -0.1810344827586207, 'best:', -0.17282608695652174, 20)
('It took', 40.91202688217163, 'seconds to complete generation')
the fit model has label_accuracy: 0.742 and domain_accuracy:0.906 and fitness_scor
e:-0.181
('Step', 57, ':', -0.16648648648648648, 'best:', -0.16648648648648648, 20)
('It took', 41.41397213935852, 'seconds to complete generation')
the fit model has label_accuracy: 0.753 and domain_accuracy:0.903 and fitness_scor
e:-0.166
('Step', 58, ':', -0.17326203208556148, 'best:', -0.16648648648648648, 20)
('It took', 42.29031991958618, 'seconds to complete generation')
the fit model has label_accuracy: 0.755 and domain_accuracy:0.913 and fitness_scor
e:-0.173
('Step', 59, ':', -0.17880085653104924, 'best:', -0.16648648648648648, 20)
('It took', 39.26325798034668, 'seconds to complete generation')
the fit model has label_accuracy: 0.749 and domain_accuracy:0.912 and fitness_scor
e:-0.179
('Step', 60, ':', -0.17330462863293863, 'best:', -0.16648648648648648, 20)
('It took', 40.730553150177, 'seconds to complete generation')
the fit model has label_accuracy: 0.750 and domain_accuracy:0.907 and fitness_scor
e:-0.173
```

In [18]:

```
saveModel('mnistmodel_20D')
```

In [19]:

```
weight_modifier.set_weights(snes.center)
```

In [31]:

```
label_classifier = Train_classifier(mnist_train,mnist_train_labels)
sourcePredictions = Predict_from_clf(label_classifier, mnist_test)
sourceAccuray = accuracy_score(mnist_test_labels,sourcePredictions)
print  "label predicitions on MNIST %0.3f" % (sourceAccuray)

targetPredictions = Predict_from_clf(label_classifier, mnistm_test)
targetAccuracy = accuracy_score(targetPredictions, mnist_test_labels)
print "label predicitons on MNISTM %0.3f" %(targetAccuracy)
```

```
label predicitions on MNIST 0.838
label predicitons on MNISTM 0.163
```

In [32]:

```
# mixing source and target data
R_MT_INDEX = np.random.choice(a=list(range(MX.shape[0])), size=MY.shape[0])
TM_X = MX[R_MT_INDEX]
TM_Y = MY[R_MT_INDEX]

TEST_MX = np.concatenate([mnist_test, mnistm_test], axis=0)
TEST_MY = np.concatenate([np.zeros(mnist_test.shape[0]), np.ones(mnistm_test.shape[0])
```

In [33]:

```
Domain_classifier = Train_classifier(TM_X, TM_Y)
domain_pred = Predict_from_clf(Domain_classifier, TEST_MX)
domain_accuracy= accuracy_score(TEST_MY, domain_pred)
print("domain predicitions accuracy: %0.3f" %(domain_accuracy))
```

```
domain predicitions accuracy: 0.946
```

In [23]:

```
import pandas as pd
```

In [29]:

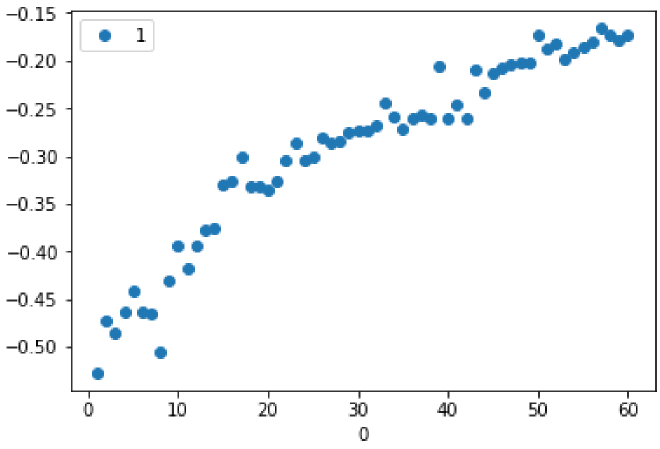
```
log = pd.read_csv('log.csv', header=None)
```

In [30]:

```
log.plot(x=0, y=1, style='o')
```

Out[30]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c32d0a990>
```


● 1
−0.15
−0.20
−0.25
−0.30
−0.35
−0.40
−0.45
−0.50
0
10
20
30
40
50
60
0