In [1]:

```
# 查看当前挂载的数据集目录
!ls /home/aistudio/data/
```

Output

```
data2156
```

In [2]:

```
import paddle
import paddle.fluid as fluid
import paddle.fluid.layers as layers
import csv
import numpy as np
import pandas

FTRAIN = '/home/aistudio/data/data2156/training.csv'
FTEST = '/home/aistudio/data/data2156/test.csv'

print 'complete'
```

Output

```
complete
```

In [3]:

```
def reader(test=False):
    # 如果test为true，表示读取测试数据
    filename = FTEST if test else FTRAIN

    if not test:
```

```
    with open(filename) as f:
        rows = csv.reader(f)
        rows.next()

        for row in rows:
            has_missing_value = False
            for cell in row:
                if not cell.strip():
                    has_missing_value = True
                    break

            # 对于有缺失值的数据不进行训练
            if has_missing_value:
                continue

            # 将最后一列数据，即Image列，转化为numpy array
            X = np.fromstring(row[-1], sep=' ')
            # 归一化处理
            X = X / 255.0
            X = X.astype(np.float32)
            # 改变形状一遍卷积网络使用
            X = np.vstack(X).reshape(1, 96, 96)
            # y为关键点坐标数据
            y = np.array(row[:-1])

            temp = []

            for i in row[:-1]:
                temp.append(float(i))

            y = np.array(temp)
            # y原本范围为[0, 95]，为了方便进行分析，将范围变成[-1, 1]
            y = (y - 48) / 48

            yield X, y
```

```
41
42  print 'complete'
43
44  def read_test():
45      result=[]
46      with open(FTEST) as f:
47          rows = csv.reader(f)
48          rows.next()
49
50          for i in range(16):
51              row = rows.next()
52              X = np.fromstring(row[1], sep=' ')
53              X = X / 255.0
54              X = X.astype(np.float32)
55              X = np.vstack(X).reshape(1, 1, 96, 96)
56              result.append(X)
57          return np.asarray(result)
58
59  train_reader=paddle.batch(paddle.reader.shuffle(reader, 2100), batch_size=42)
60  print 'complete'
```

Output

```
complete
complete
```

In [4]:

```
1  def predict_function():
2      #input层，输入的shape有1个channel，大小为96*96
3      input_data=layers.data(name='input_data', shape=[1, 96, 96], dtype='float
4
5      #第1个卷积层，filter数量为32，大小3*3；池化层大小2*2，步长为2
6      convolutional_1 = fluid.nets.simple_img_conv_pool(
7          input=input_data,
8          num_filters=32,
9          filter_size=3,
```

```
        pool_size=2,
        pool_stride=2,
        act='relu'
    )

    #第2个卷积层，filter数量为64，大小2*2；池化层大小2*2，步长为2
    convolutional_2 = fluid.nets.simple_img_conv_pool(
        input=convolutional_1,
        num_filters=64,
        filter_size=2,
        pool_size=2,
        pool_stride=2,
        act='relu'
    )

    #第3个卷积层，filter数量为128，大小2*2；池化层大小2*2，步长为1
    convolutional_3 = fluid.nets.simple_img_conv_pool(
        input=convolutional_2,
        num_filters=128,
        filter_size=2,
        pool_size=2,
        pool_stride=1,
        act='relu'
    )

    #第1隐层，unit数量为500
    hidden_1 = layers.fc(
        input=convolutional_3,
        size=500,
        act='relu'
    )

    #第2隐层，unit数量为500
    hidden_2 = layers.fc(
        input=hidden_1
```

```
        input=hidden_1,
        size=500,
        act='relu'
    )

    prediction = layers.fc(
        input=hidden_2,
        size=30
    )

    return prediction


def training():
    prediction = predict_function()

    original_mark = layers.data(
        name='original_mark',
        shape=[30],
        dtype='float32'
    )

    loss = layers.square_error_cost(input=prediction, label=original_mark)

    #损失函数为MSE
    return layers.mean(loss)


def optimization():
    #在比较各种优化方法后，nesterov_momentum为下降速度最快的方法
    return fluid.optimizer.Momentum(learning_rate=0.01, momentum=0.9)

print 'complete'
```

Output

```
complete
```

complete

In [5]:

```
from paddle.v2.plot import Ploter

title='MSE Cost'
plot_cost=Ploter(title)

step=0
params_dirname = "keypoint.inference.model"
def event_handler(event):
    global step
    if isinstance(event, fluid.EndStepEvent):
        print event.epoch
        print event.step
        print event.metrics[0]
        plot_cost.append(title, step, event.metrics[0])
        plot_cost.plot()

        #保存训练模型
        if params_dirname is not None:
            trainer.save_params(params_dirname)

    step += 1

print 'complete'
```

Output

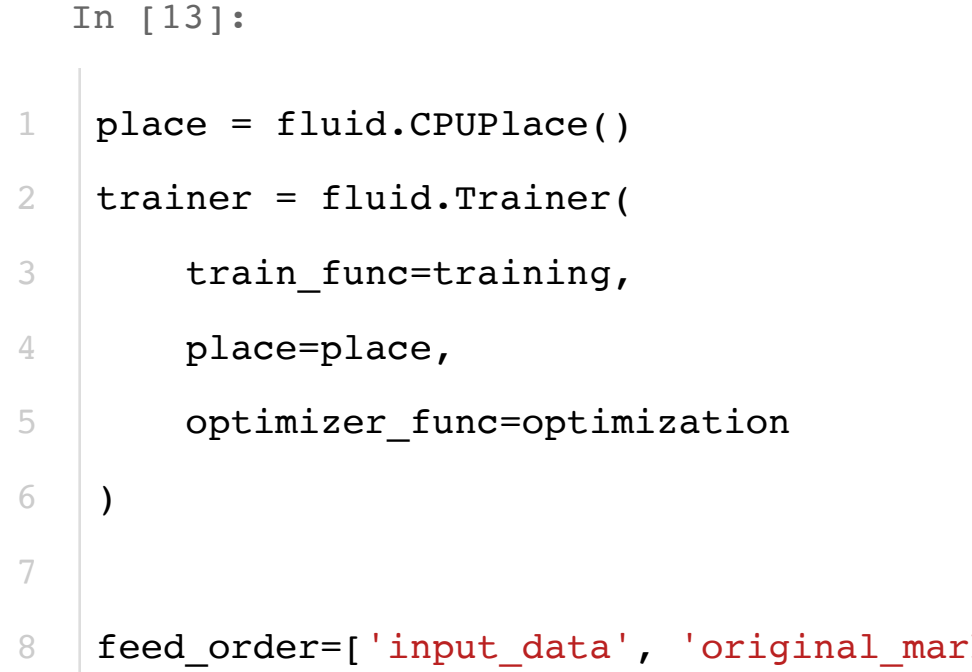
```
complete
```

In [13]:

```
place = fluid.CPUPlace()
trainer = fluid.Trainer(
    train_func=training,
    place=place,
    optimizer_func=optimization
)

feed_order=['input_data', 'original_mark']
trainer.train(
    reader=train_reader,
    num_epochs=2,
    event_handler=event_handler,
    feed_order=feed_order
)

print 'complete'
```



Output

```
complete

<Figure size 432x288 with 0 Axes>
```

In [6]:

```
inferencer = fluid.Inferencer(
    infer_func=predict_function,
    param_path=params_dirname,
    place=fluid.CPUPlace()
)

test = read_test()

predicts = []
for i in range(16):
    # 一次只能feed一个sample
    predict=inferencer.infer({'input_data': test[i]})
    predict_np = np.asarray(predict).reshape(30)
    predicts.append(predict_np)

predicts = np.asarray(predicts)


print 'complete'
```

Output

```
complete
```

In [7]:

```
from matplotlib import pyplot

def plot_sample(x, y, axis):
    img = x.reshape(96, 96)
    axis.imshow(img, cmap='gray')
    axis.scatter(y[0::2] * 48 + 48, y[1::2] * 48 + 48, marker='x', s=10)

fig = pyplot.figure(figsize=(6, 6))
fig.subplots_adjust(
    left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05)

for i in range(16):
    ax = fig.add_subplot(4, 4, i + 1, xticks=[], yticks=[])
    plot_sample(test[i], predicts[i], ax)

pyplot.show()
```

In [8]:

运行 上移 下移 删除

```
# 查看个人持久化工作区文件
!ls /home/aistudio/work/
```