Using lstm.py from OCRopus

This worksheet illustrates how to use the `lstm.py` LSTM implementation in the OCRopus OCR system.

In [1]:
```
from pylab import *
import random as pyrandom
import lstm
from functools import partial
```

We now create a 1D single directional LSTM with 2 inputs, 2 memory cells, and 1 output.

`lstm.LSTM1` creates the simplest kind of LSTM. Note that for most applications, you probably want to use `lstm.BIDILSTM` instead.

In [2]:
```
net = lstm.LSTM1(2,2,1)
net.setLearningRate(1e-1,0.0)
```

Here is a simple test case generator. It generates an input sequence and a target sequence of the same length.

In [3]:
```
def generate_mod(n=50,ninput=2,m=3,example=0):
    "Generate a regular beat every m steps. The input is random."
    x = rand(n,ninput)
    y = 1.0*(arange(n,dtype='i')%m==0).reshape(n,1)
    return x,y
```

Let's generate a simple test case. Note that this generates a 50x2 array for the input and a 50x1 array for the output.

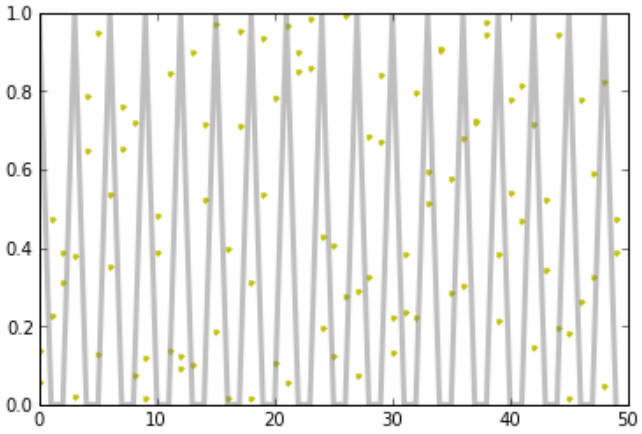
In [4]:
```
xs,ys = generate_mod(n=50)

print xs.shape,ys.shape
```

```
(50, 2) (50, 1)
```

Let's look at what the input and output look like.

In [5]:
```
plot(xs,"y.")
plot(ys,color='gray',alpha=0.5,linewidth=3)
```

Out[5]: [<matplotlib.lines.Line2D at 0xaf43b6c>]



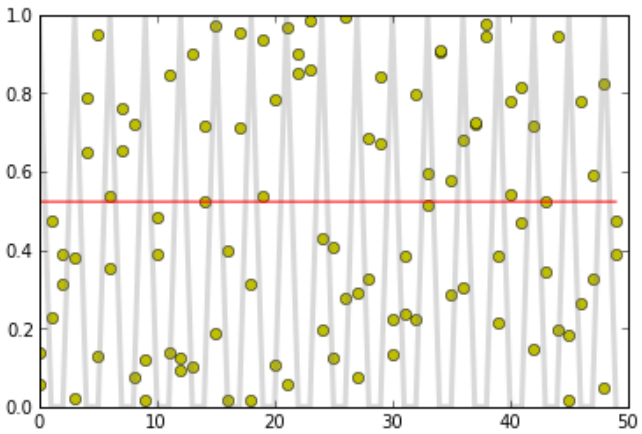
Trainnnig is very simple: we give the network the input and corresponding output sequence.

In [6]:
```
preds = net.train(xs,ys)
```

The output isn't very informative yet (not much training).

In [7]:
```
plot(xs,"yo")
plot(ys,color='gray',alpha=0.3,linewidth=3)
plot(array(preds)[:,0],'r')
```

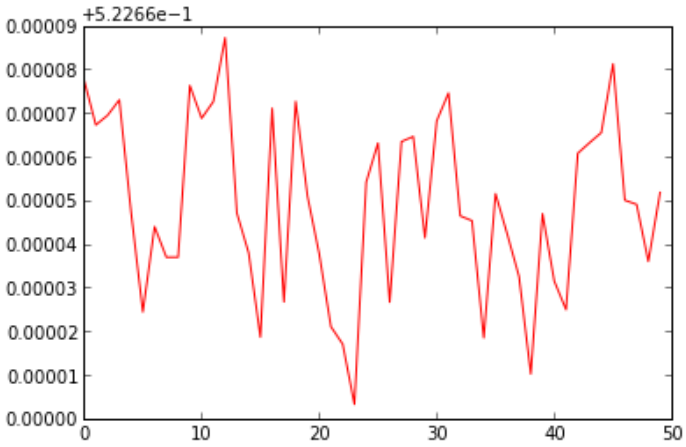
Out[7]: [<matplotlib.lines.Line2D at 0xaf64c4c>]



Although the output looks constant, it actually isn't. It's just that the activations are small right now, given the small random weights we initialized the network with.

In [8]:
```
plot(array(preds)[:,0],'r')
```

Out[8]:
```
[<matplotlib.lines.Line2D at 0xb11946c>]
```



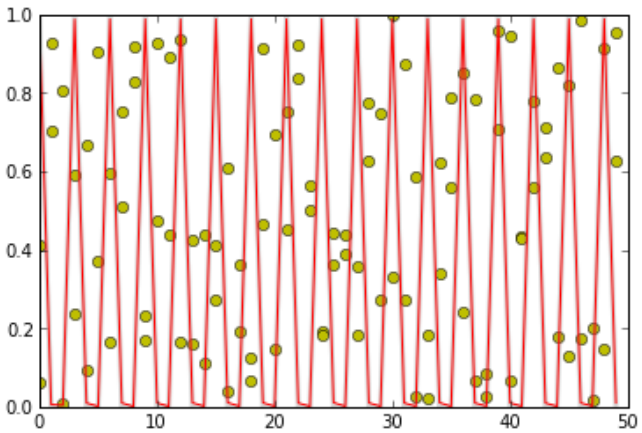
Let's train this a lot more. Since the input doesn't matter, let's just keep using the same training example.

In [9]:
```
for i in range(10000):
    xs,ys = generate_mod(n=50)
    preds = net.train(xs,ys)
```

So what does the output look like now?

In [10]:
```
plot(xs,"yo")
plot(ys,color='gray',alpha=0.3,linewidth=3)
plot(array(preds)[:,0],'r')
```

Out[10]:
```
[<matplotlib.lines.Line2D at 0xb168fec>]
```



Looks like the training worked well. You can see that the red line is the output from the network because it's not entirely perfect (look after the very first peak, where it doesn't drop to zero).