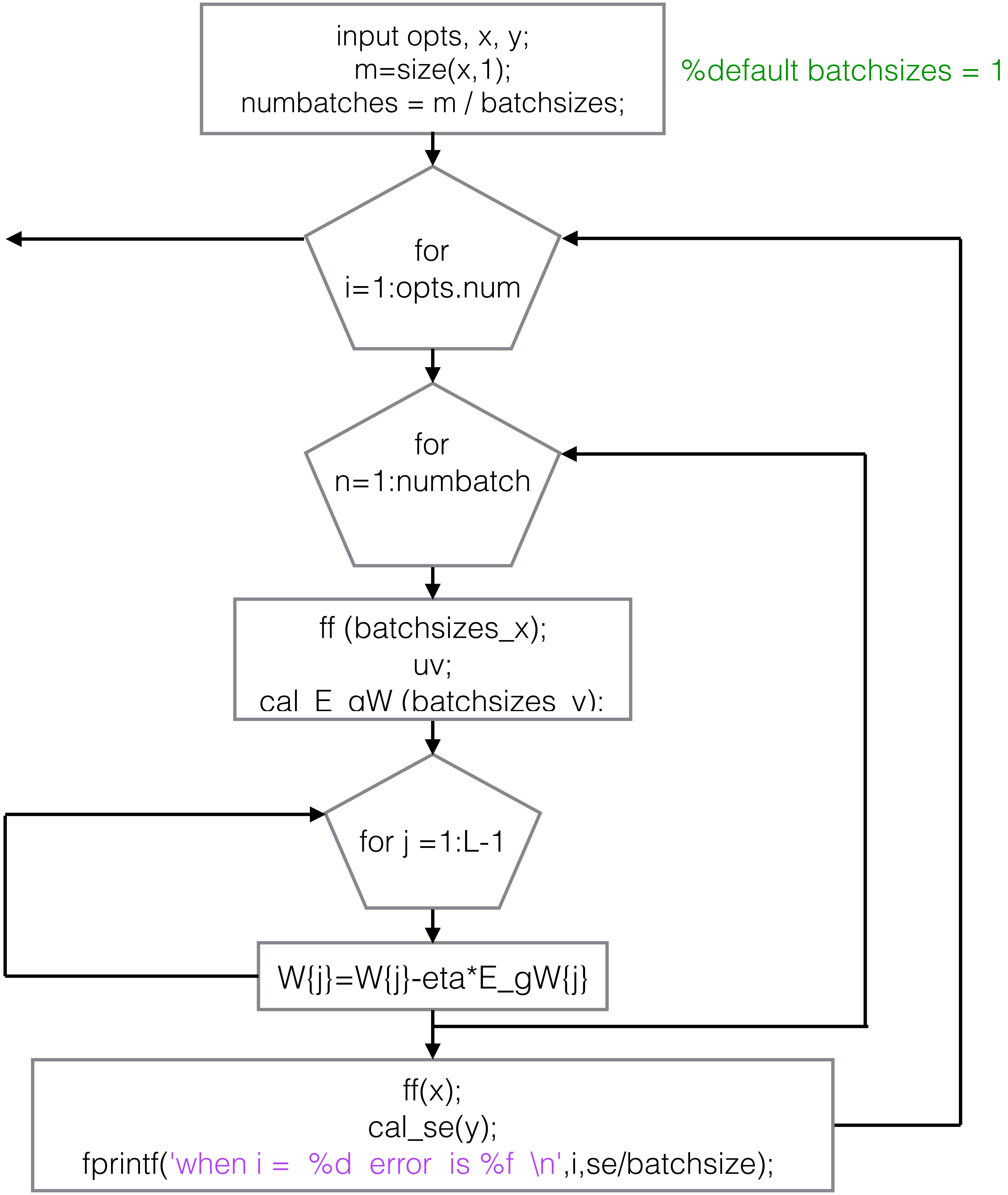
```
% define a net of deep perceptrons
 w{1}=[1 1 2;1 -1 -1]';nL{1}='tanh';
 w{2}=[1 -1 1;-1 -1 -1; -2 1.5 0.5]';nL{2}='tanh';
 w{3}=[1 1 1/2 1;1 -1 1 -1]';nL{3}='tanh';
 layers=4;
 net=perceptrons(layers,w,nL);
% draw network functions if input dim is 2
net.draw();
x=rand(300,2)*10-5;
x_test=rand(300,2)*10-5;
net=net.ff(x);
y=net.a{layers};
net=net.ff(x_test);
y_test=net.a{layers};
plot3(x(:,1),x(:,2),y,'.')
net=net.cal_uv(); % backpropagation
% consider y as desired output
% randomize weight matrices
for i=1:layers-1
    W{i}=rand(size(w{i}));
end
net2=perceptrons(layers,W,nL);
net2=net2.ff(x);
y_hat=net2.a{layers};
net2=net2.cal_uv(); % backpropagation
net2=net2.cal_E_gW(y);
err_g=net2.gradient_check2(x,y);

opts.num = ;  %numepochs
opts.batchsize = ;   %batchsize
opts.eta = ;    %learning rate
net2=net2.train(opts,x,y);
net2=net2.test(x_test,y_test);
figure
net2.draw();
```

Add method train

Add method test

function train



input opts, x, y;
m=size(x,1);
numbatches = m / batchsizes;

%default batchsizes = 1

for
i=1:opts.num

for
n=1:numbatch

ff (batchsizes_x);
uv;
cal_E_gW (batchsizes_y);

for j =1:L-1

W{j}=W{j}-eta*E_gW{j}

ff(x);
cal_se(y);
fprintf('when i = %d error is %f \n',i,se/batchsize);

Exercise 1

```
% define a net of deep perceptrons
 w{1}=[1 1 2;1 -1 -1]';nL{1}='tanh';
 w{2}=[1 -1 1;-1 -1 -1; -2 1.5 0.5]';nL{2}='tanh';
 layers=3;
 net=perceptrons(layers,w,nL);
% draw network functions if input dim is 2
net.draw();
x=rand(300,2)*10-5;
x_test=rand(300,2)*10-5;
net=net.ff(x);
y=net.a{layers};
net=net.ff(x_test);
y_test=net.a{layers};
plot3(x(:,1),x(:,2),y,'.')
net=net.cal_uv(); % backpropagation
% consider y as desired output
% randomize weight matrices
for i=1:layers-1
    W{i}=rand(size(w{i}));
end
net2=perceptrons(layers,W,nL);
net2=net2.ff(x);
y_hat=net2.a{layers};
net2=net2.cal_uv(); % backpropagation
net2=net2.cal_E_gW(y);
err_g=net2.gradient_check2(x,y);

opts.num = ;  %numepochs
opts.batchsize = ;  %batchsize
opts.eta = ;   %learning rate
net2=net2.train(opts,x,y);
net2=net2.test(x_test,y_test);
figure
net2.draw();
```

change your net from layers = 4 to layers = 3

```
  when i = 1 error is 4.174634
  when i = 2 error is 2.895957
  when i = 3 error is 1.019896
  when i = 4 error is 0.453433
  when i = 5 error is 0.298348
  when i = 6 error is 0.226201
  when i = 7 error is 0.182881
                .
                .
                .
when i = 7991 error is 0.000006
when i = 7992 error is 0.000006
when i = 7993 error is 0.000006
when i = 7994 error is 0.000006
when i = 7995 error is 0.000006
when i = 7996 error is 0.000006
when i = 7997 error is 0.000006
when i = 7998 error is 0.000006
when i = 7999 error is 0.000006
when i = 8000 error is 0.000006
>>
```

Exercise 2

```
% define a net of deep perceptrons
 w{1}=[1 1 2;1 -1 -1]';nL{1}='tanh';
 w{2}=[1 -1 1;-1 -1 -1; -2 1.5 0.5]';nL{2}='tanh';
 w{3}=[1 1 1/2 1;1 -1 1 -1;1 1 -2 1]';nL{3}='tanh';
 w{4}=[1 1 1/2 1;0.5 2 -1 1]';nL{4}='tanh';
 layers=5;
 net=perceptrons(layers,w,nL);
% draw network functions if input dim is 2
net.draw();
x=rand(300,2)*10-5;
x_test=rand(300,2)*10-5;
net=net.ff(x);
y=net.a{layers};
net=net.ff(x_test);
y_test=net.a{layers};
plot3(x(:,1),x(:,2),y,'.')
net=net.cal_uv(); % backpropagation
% consider y as desired output
% randomize weight matrices
for i=1:layers-1
    W{i}=rand(size(w{i}));
end
net2=perceptrons(layers,W,nL);
net2=net2.ffx);
y_hat=net2.a{layers};
net2=net2.cal_uv(); % backpropagation
net2=net2.cal_E_gW(y);
err_g=net2.gradient_check2(x,y);
opts.num = ;  %numepochs
opts.batchsize = ;   %batchsize
opts.eta = ;    %learning rate
net2=net2.train(opts,x,y);
net2=net2.test(x_test,y_test);
figure
net2.draw();
```

change your net from layers = 4 to layers = 5

```
  when i = 1 err is 0.705183
  when i = 2 err is 0.668718
  when i = 3 err is 0.651212
  when i = 4 err is 0.635701
  when i = 5 err is 0.618218
  when i = 6 err is 0.596136
  when i = 7 err is 0.566321
  when i = 8 err is 0.524401
  when i = 9 err is 0.465395
 when i = 10 err is 0.390159
               .
               .
               .
when i = 7990 err is 0.000246
when i = 7991 err is 0.000245
when i = 7992 err is 0.000246
when i = 7993 err is 0.000245
when i = 7994 err is 0.000246
when i = 7995 err is 0.000245
when i = 7996 err is 0.000246
when i = 7997 err is 0.000245
when i = 7998 err is 0.000245
when i = 7999 err is 0.000245
when i = 8000 err is 0.000245
>>
```

Exercise 3

```
%load data
load mnist_uint8.mat
train_x = double(train_x) / 255;
test_x = double(test_x) / 255;
train_y = double(train_y);
test_y = double(test_y);
% normalize
[train_x, mu, sigma] = zscore(train_x);
test_x = normalize(test_x, mu, sigma);
% w
w{1} = rand(785,100);nL{1}='tanh';
w{2} = rand(101,10);nL{2}='tanh';

layers=3;

net = perceptrons(layers,w,nL);
net = net.ff(train_x);
y_hat = net.a{layers};

net = net.uv(); % backpropagation
net = net.cal_E_gW(train_y);

opts.num = ;
opts.eta = ;
opts.batchsize = ;

net=net.train(opts,train_x,train_y);
net=net.test(test_x,test_y);
```

Use your net to learn mnist_uint8 data