Contents

Stochastic Gradient descent

This is the script for batch gradient descent algorithm using polynomial base function

```
clc;
clear;
close all;
```

Generate data: generate t0 from x0

```
x=[0.0005:0.0005:1.000]';
noise=0.1.*randn(1,2000)';
t=sin(2*pi*x)+noise;
% plot(x,t);
% hold on
```

Split data into train set and test set

```
idx=crossvalind('Kfold',size(x,1),5);
test_idx = find(idx==1);
train_idx = find(idx~=1);
train_x=x(train_idx,:);
train_t=t(train_idx,:);
test_x=x(test_idx,:);
test_t=t(test_idx,:);
```

Training params

Below is the equation for Stochastic Gradient Descent. Here j is the power of x coresponding to w_j, n iterates through all data points. η is the learning rate for each iteration. $W_j^{n-1} = W_j^n - \eta(y(x_n, W) - t_n) * x^j)$

```
M=4; % Maximum power of x
max_iter=5000000;
eta=0.9;
W=zeros(M+1,1);
W_threshold = repmat(0.000000001, M+1, 1);
err_threshold = .000000001;
```

Training

Here comes the base matrix with power 4

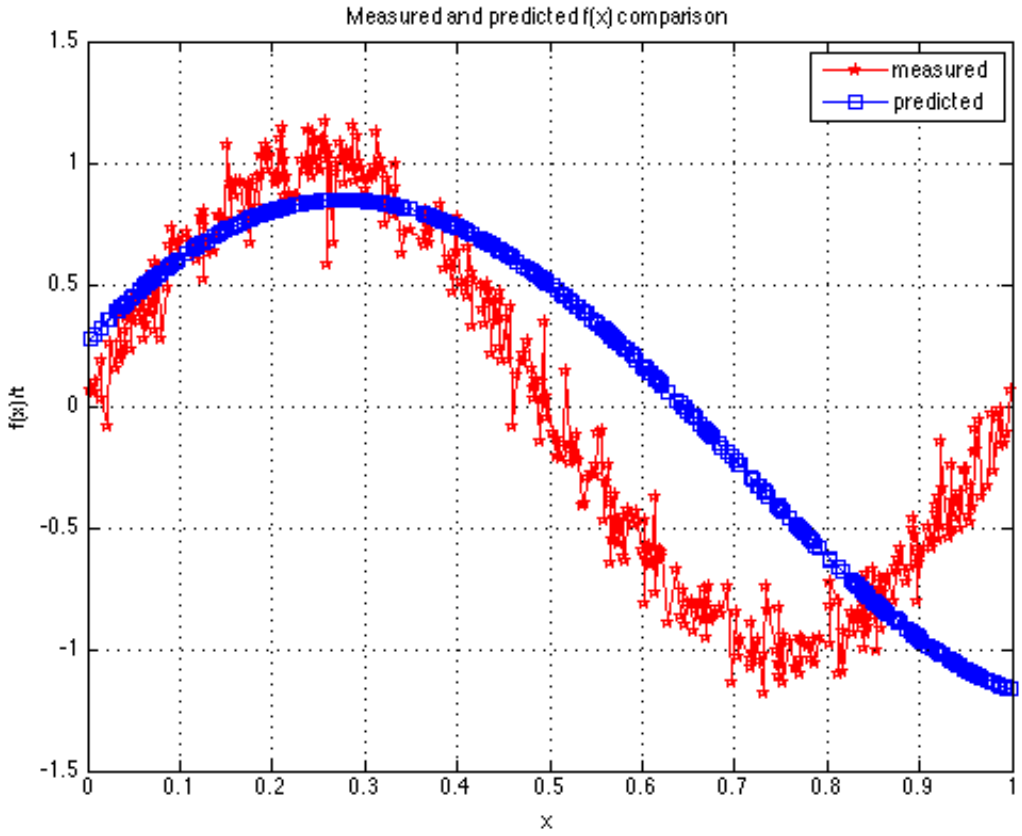
```
x_pol=[ones(size(train_x,1),1) train_x train_x.*train_x train_x.*train_x.*train_x ...
    train_x.*train_x.*train_x.*train_x];
% size(x_pol)
for iter = 1:max_iter
    W_prev = W;
    % Enable using data more than once, idx_crt is current index of the point
    idx_crt = rem(iter-1,size(train_x,1))+1;
    fx_n = (W'*x_pol(idx_crt,:)')';
    W = W - eta*1/size(train_x,1)*((fx_n - t(idx_crt,:))*x_pol(idx_crt,:))';
%     W = W - eta*1/size(train_x,1)*((fx_n - t(idx_crt,:))*x_pol(idx_crt,:))';

    % Sum square error on training
    y_hat=(W'*x_pol')';
    err(iter,1) = 1/2*sum((y_hat - train_t).^2);
    % check for termination conditions
    if iter>1 && abs(err(iter-1,1)-err(iter,1))< err_threshold
        break;
    end
end
```

Testing

```
pred_fx=(W'*[ones(size(test_x,1),1) test_x test_x.*test_x test_x.*test_x.*test_x ...
    test_x.*test_x.*test_x.*test_x]')';
test_err = 1/2*sum((pred_fx - t(test_idx,:)).^2);
rmse = sqrt(2*test_err/size(test_idx,1));
```

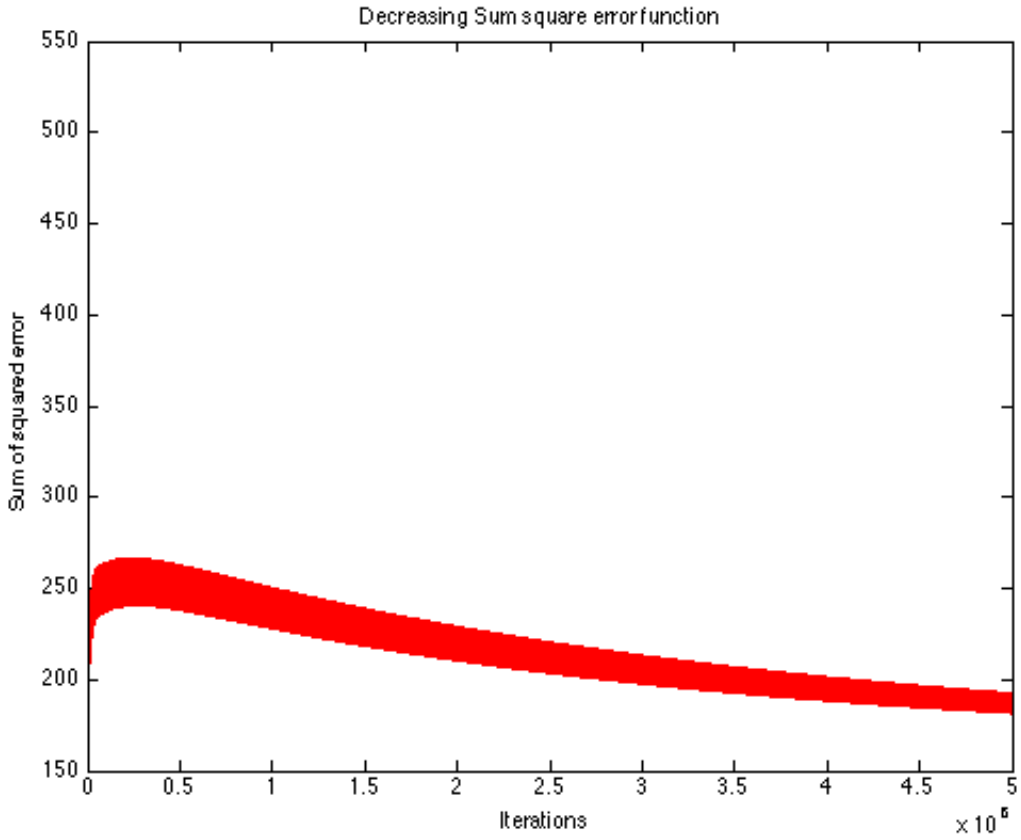
Results 1: Figure for predicted and actual comparison

```
fig1=figure(1);
plot(x(test_idx),t(test_idx),'rp-',x(test_idx),pred_fx,'bs-');
legend('measured', 'predicted');
grid on;
xlabel('x');
ylabel('f(x)/t');
title('Measured and predicted f(x) comparison');
saveas(fig1,'ms_pred_cmp.jpg','jpg');
```



Result 2: Error curve with different iterations

```
fig2=figure(2);
plot(err,'r');
title('Decreasing Sum square error function');
xlabel('Iterations');
ylabel('Sum of squared error');
saveas(fig2,'Sum_sqr_err.jpg','jpg');
```



Result 3: Final Root-Mean-Square Error and W

```
fprintf('\nRoot mean square error: ');
rmse
fprintf('\ncoefficient W:\n');
W
```

```
Root mean square error:
rmse =

      0.47051


coefficient W:

W =

      0.26028
       4.2105
      -7.0408
      -3.0885
       4.4926
```

Published with MATLAB® R2013a