Table of Contents

```
clear all
close all
```

Part 1 - Retrieval

```
load("mnist.mat")
%
% figure(1)
% clf
% i = 1;
% imshow(reshape(trainX(i,:),28,28)')
% title(trainY(i))
```

Part 2 - Pre-processing

```
idx = trainY == 4 | trainY == 9;
Atr = double(trainX(idx,:));
btr = double(trainY(idx))';
ntr = size(Atr, 2);
mtr = size(Atr, 1);
btr(btr==4)=1;
btr(btr==9)=-1;

idx_test = testY == 4 | testY == 9;
Atest = double(testX(idx_test,:));
btest = double(testY(idx_test))';
mtest = size(Atest, 1);
% Turn labels into +1 -1
btest(btest==4)=1;
btest(btest==9)=-1;

% Normalization
[Atr, Amean, Astd] = normalize(Atr);
% NOTE: It is important to not re-compute the Amean and Astd for 2
 reasons:
```

```
% 1. The training set has 10x more samples and the mean and standard
% deviation are more representative of the population mean
% 2. A rule of thumb when evaluating the performance of your model is
 to
% avoid touching the test set to avoid overtrainning

Atest = Atest - ones(mtest,1)*Amean;
Atest = Atest ./ max(ones(mtest,1)*Astd,1);
% Validation Functions
C = @(z) (z > 0)*2 - 1;
I = @(x,y) x ~= y;
misclass_rate = @(A,y,x) sum(I(C(A*x), y))/length(y);
```

Part 3 - Linear Regression

```
btr_ls = btr;
btest_ls = btest;
x_lr = Atr \ btr;

train_loss = norm(Atr*x_lr - btr_ls, 2)
test_loss = norm(Atest*x_lr - btest_ls, 2)

train_misclass_rate_lr = misclass_rate(Atr, btr_ls, x_lr)
test_misclass_rate_lr = misclass_rate(Atest, btest_ls, x_lr)
```

```
Warning: Rank deficient, rank = 607, tol =  2.842930e-10.

train_loss =

   46.2200


test_loss =

   26.3921


train_misclass_rate_lr =

    0.0308


test_misclass_rate_lr =

    0.0362
```

Part 4 - Logistic Regression

Pre-process

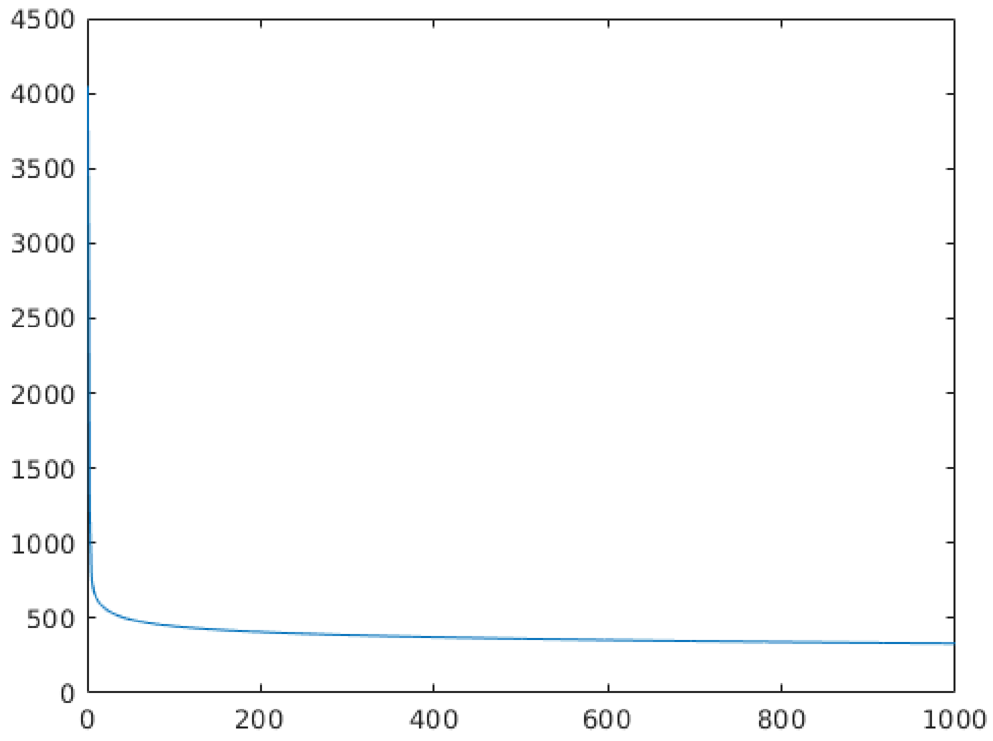
```
btr = (btr+1)/2;
```

```
btest = (btest+1)/2;

% Initialize functions
sig = @(x) 1.0 ./(1+exp(-x));
f = @(x) f_func(Atr, btr, x, sig);
g = @(x) Atr'*(sig(Atr*x) -btr);
x0 = zeros(ntr, 1);
epsilon = 1e-4;
max_iter = 1e3;
```

Gradient Descent

```
[x_gd, trace_gd, status] = gd(f, g, x0, 1/mtr ,max_iter, epsilon);
if status < 0
    disp("GD diverged")
end
train_misclass_rate_gd = misclass_rate(Atr, btr,  x_gd)
test_misclass_rate_gd = misclass_rate(Atest, btest, x_gd )
figure(1)
plot(trace_gd)
hold on
```

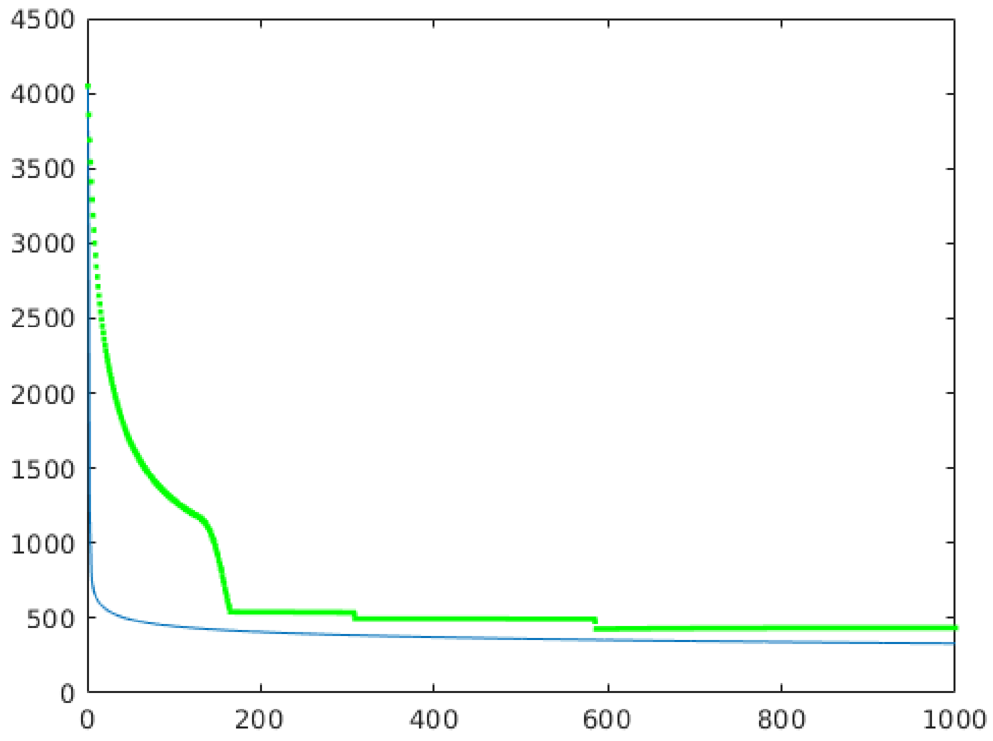
```
train_misclass_rate_gd =

    0.5146


test_misclass_rate_gd =

    0.5239
```



Gradient Descent With Backtracking Line Search

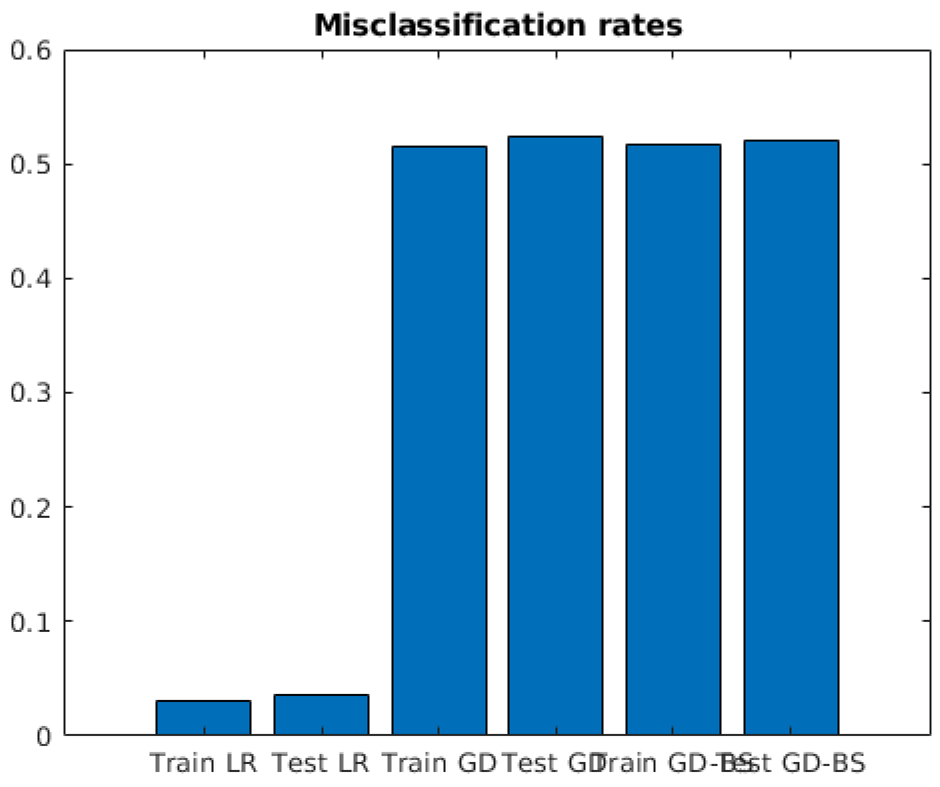
```
[x_gd_bt, trace_bt, status] = gd_bt(f, g, x0, 1, 0.5, 0.5, 1000,
 1e-1);
if status < 0
    disp("GD diverged")
end
train_misclass_rate_gd_btls = misclass_rate(Atr, btr, x_gd_bt)
test_misclass_rate_gd_btls = misclass_rate(Atest, btest, x_gd_bt)
plot(trace_bt, 'g.')
hold off
```

```
train_misclass_rate_gd_btls =

    0.5171


test_misclass_rate_gd_btls =

    0.5203
```



Comparisons

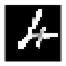
```
figure(2)
rates = [train_misclass_rate_lr, test_misclass_rate_lr,
 train_misclass_rate_gd,test_misclass_rate_gd,
 train_misclass_rate_gd_btls, test_misclass_rate_gd_btls];
bar(rates)
title('Misclassification rates')
set(gca,'xticklabel',{'Train LR', 'Test LR', 'Train GD','Test
 GD','Train GD-BS', 'Test GD-BS'});
```



Debugging

Linear regression worst samples

```
[M, ind] = maxk(abs(Atest*x_lr - btest_ls),3 );
figure(3)
imshow(reshape(Atest(ind(1),:),28,28)');
figure(4)
imshow(reshape(Atest(ind(2),:),28,28)');
figure(5)
imshow(reshape(Atest(ind(3),:),28,28)');
% Logistic regression worst samples
sig(Atest*x_gd);
[M, ind] = maxk(abs(sig(Atest*x_gd) - btest),3);
figure(6)
imshow(reshape(Atest(ind(1),:),28,28)');
figure(7)
imshow(reshape(Atest(ind(2),:),28,28)');
figure(8)
imshow(reshape(Atest(ind(3),:),28,28)');
```



Helper Functions

```
function [X, avg, Xstd] = normalize(X)
    [m, ~] = size(X);
    avg = mean(X,1);
    X = X - ones(m,1)*avg;
    Xstd = std(X,1);
    X = X ./ max(ones(m,1)*Xstd,1);
end

function cost = f_func(A, b, x, act_func)
    m = length(b);
    z = act_func(A*x);
    cost = sum(-log(z(b == 1))) + sum(-log(1 - z(b == 0)))/m;
end
```

Published with MATLAB® R2018a