HW8

Carl Mueller
CSCI 5254 - Convex Optimization

May 2, 2018

8.16

a)

Gradient Method

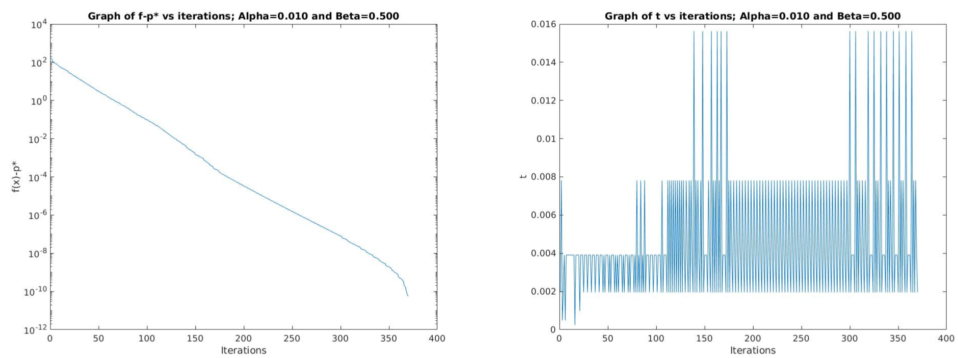

Figure 1: ALPHA=.01, BETA=.5

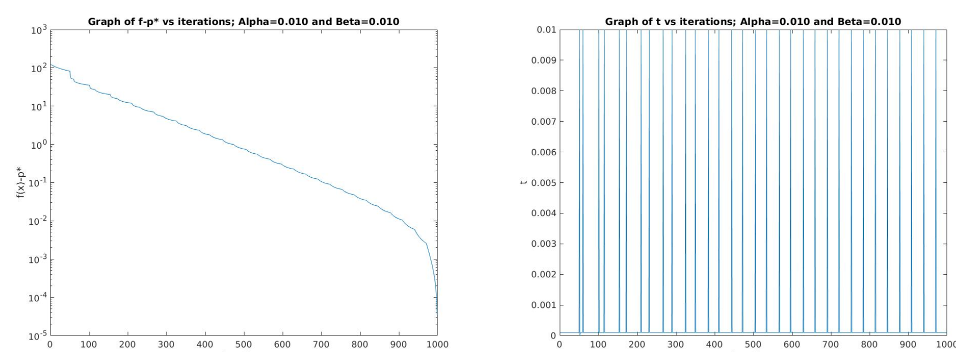

Figure 2: ALPHA=.01, BETA=.01

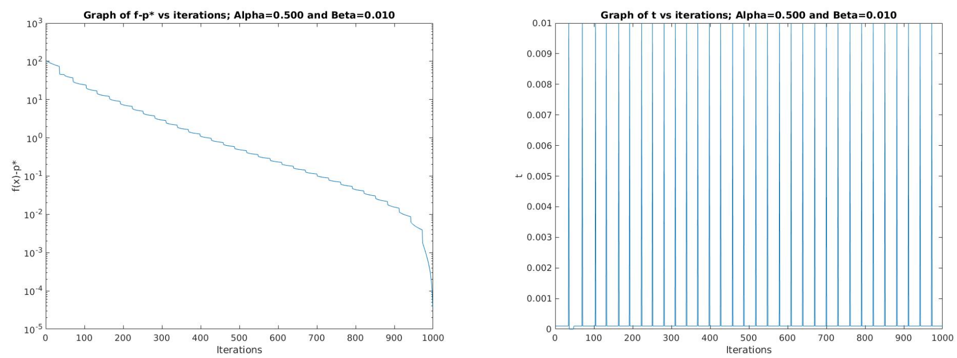

Figure 3: ALPHA=.50, BETA=.01

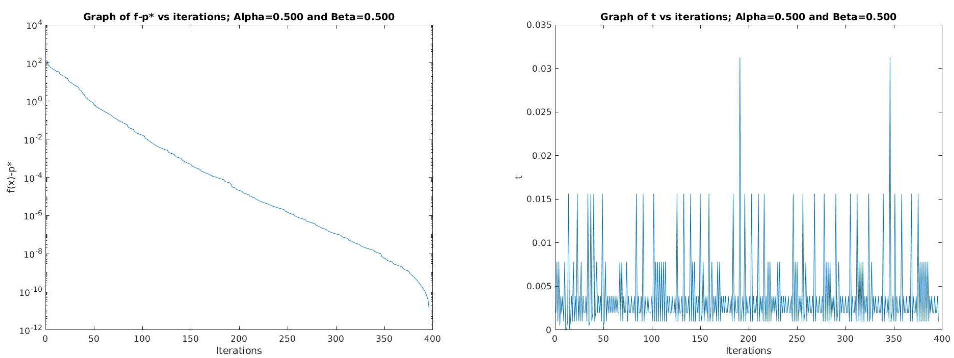

Figure 4: ALPHA=.50, BETA=.50

```
iter = 1000;
nu = .0001;
beta = .01;
alpha = .5;
n = 100;
m = 200;
x = zeros(n, 1);
A = randn(m,n);

V = []
I = []
T = []
for i = 1:iter
    % function evaluation
    f = -sum(log(1-A*x))-sum(log(1+x)) - sum(log(1-x));
    % gradient
    grad = A'*(1./(1-A*x)) - 1./(1+x) + 1./(1-x);
    % breaking criterion using nu as threshold
```

```
    if norm(grad) < nu
        break
    end
    % Gradient direction.
    dir = -grad;
    % second compoenent of backtracking
    fprime = grad'*dir;
    t = 1;
    while ((max(A*(x+t*dir)) >= 1) || (max(abs(x+t*dir)) >= 1))
        t = beta*t;
    end
    % backtracking algorithm
    while ( -sum(log(1-A*(x+t*dir))) - sum(log(1-(x+t*dir).^2)) > f + alpha*t
        t = beta*t;
    end
    % update step
    x = x+t*dir;
    T = [T; t]
    V = [V; f];
    I = [I ; i]
end

f_minus_p = [];
for i = 1:length(V)
    diff = V(i) - f
    f_minus_p = [f_minus_p; diff]
end
f_minus_p
f
figure(1)
plot(I,f_minus_p);
set(gca,'yscale','log');
titlestr = "Graph of f-p* vs iterations; Alpha=%0.3f and Beta=%0.3f";
str = sprintf(titlestr,alpha,beta);
title(str);
xlabel("Iterations");
ylabel("f(x)-p*");

figure(2)
plot(I,T);
titlestr = "Graph of t vs iterations; Alpha=%0.3f and Beta=%0.3f";
str = sprintf(titlestr,alpha,beta);
title(str);
xlabel("Iterations");
ylabel("t");
```

b)

Newton's Method

This approach clearly takes many less iterations and is always terminated based on the quit criteria rather than the max number of iterations.

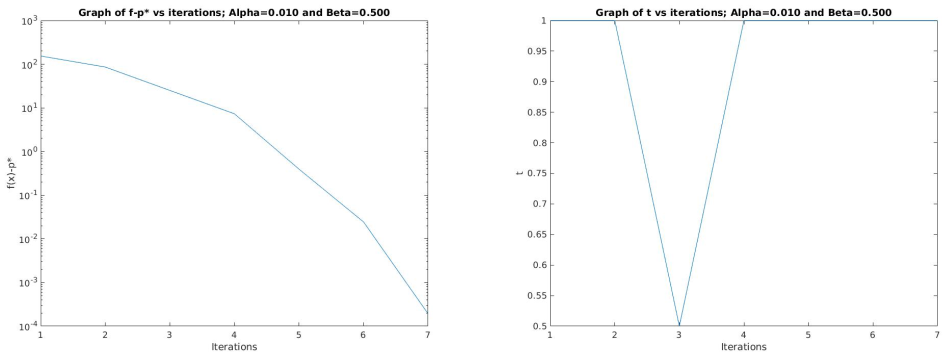

Figure 5: ALPHA=.01, BETA=.5

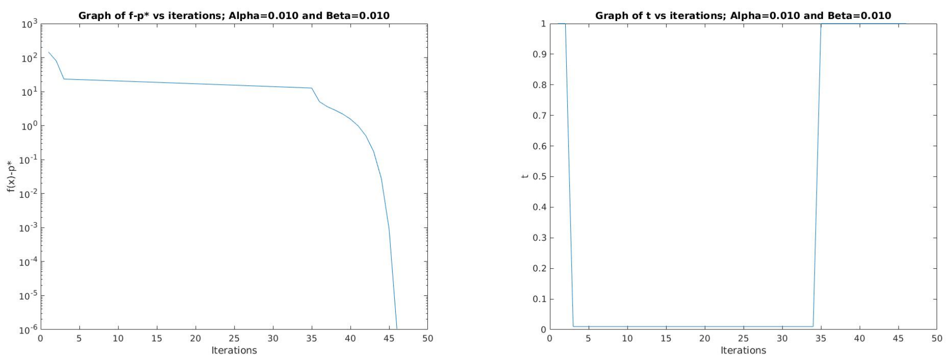

Figure 6: ALPHA=.01, BETA=.01

```
iter = 1000;
nu = .00000001;
beta = .01;
alpha = .50;
n = 100;
m = 200;
x = zeros(n, 1);
A = randn(m,n);


x = zeros(n,1);
```

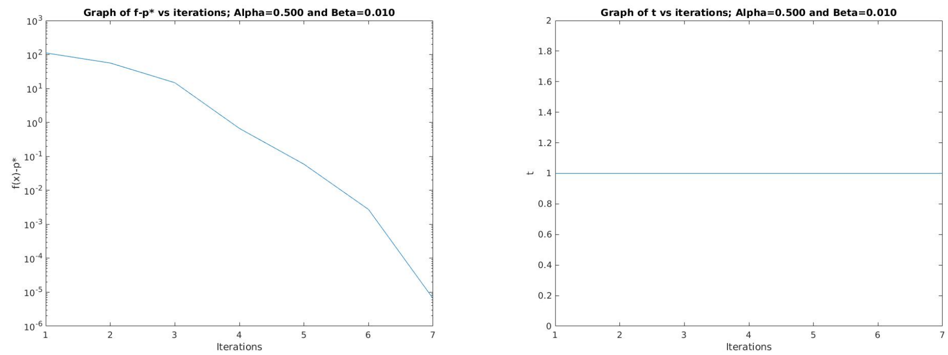

Figure 7: ALPHA=.50, BETA=.01

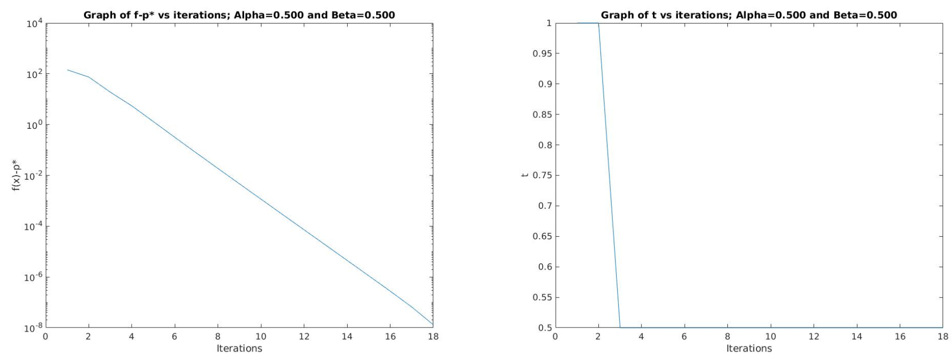

Figure 8: ALPHA=.50, BETA=.50

```
V = []
I = []
T = []
for i = 1:iter
    f = -sum(log(1-A*x)) - sum(log(1+x)) - sum(log(1-x));
    d = 1./(1-A*x);
    % first order derivative
    grad = A'*d - 1./(1+x) + 1./(1-x);
    % second order derivative i.e. hessian
    hessian = A'*diag(d.^2)*A + diag(1./(1+x).^2 + 1./(1-x).^2);
    % direction
    dir = -hessian\grad;
    % lambda^2 i.e decrement
    lambda_2 = grad'*dir;
    t = 1;
    while ((max(A*(x+t*dir)) >= 1) || (max(abs(x+t*dir)) >= 1))
        t = beta*t;
    end
```

```
    % breaking criteria
    if abs(lambda_2) < nu
        break;
    end
    % backtracking algorithm
    while ( -sum(log(1-A*(x+t*dir))) - sum(log(1-(x+t*dir).^2)) > f + alpha*t
        t = beta*t;
    end
    % update step
    x = x+t*dir;
    V = [V; f];
    I = [I ; i];
    T = [T; t];
end

f_minus_p = [];
for i = 1:length(V)
    diff = V(i) - f
    f_minus_p = [f_minus_p; diff]
end
f_minus_p
f
figure(1)
semilogy(I, f_minus_p)
titlestr = "Graph of f-p* vs iterations; Alpha=%0.3f and Beta=%0.3f";
str = sprintf(titlestr,alpha,beta);
title(str);
xlabel("Iterations");
ylabel("f(x)-p*");

figure(2)
plot(I,T);
titlestr = "Graph of t vs iterations; Alpha=%0.3f and Beta=%0.3f";
str = sprintf(titlestr,alpha,beta);
title(str);
xlabel("Iterations");
ylabel("t");
```

9.31