Table of Contents

runGLMUCB.m

```
% Runs GLM-UCB algorithm and returns regret and fraction of pulls.
%
% This code runs the OFUL algorithm adapted to be used in our setting.
% For more information, on OFUL algorithm see the original paper:
%
% -https://papers.nips.cc/paper/4166-parametric-bandits-the-
generalized-linear-case.
```

Inputs:

```
k: Number of arms.
T: Time horizon.
d: Dimension of covariates.
b: A k*d matrix of arm parameters.
xmax: Maximum of l2-norm of covariates
    (used only for context generation). This parameter is unused if
    noise and contexts are provided.
delta: The probability that confidence intervals fail.
contexts: A T*d matrix of contexts.
reward_vector: A T*k matrix of reward vectors.
verbose: Whether to print outputs or not.
```

Outputs:

```
regret: Cumulative regret as a running sum over regret terms.
fractions: Fractions of pulls of different arms.
```

Code:

```
function [regret, fractions] = runGLMUCB(k, T, d, b, xmax, ...
    delta, contexts, reward_vector, verbose)

pull_ind = zeros(T,k); % indicator whether each arm is pulled at each
 round t

regret = zeros(1,T);

betahat = b*0;
```

```
Vt = zeros(k*d, k*d);

max_norm_b = 0;

for i=1:k
    tmp_norm = norm(b(i,:),2);
    if(tmp_norm > max_norm_b);
        max_norm_b = tmp_norm;
    end
end

% Parameters of GLM-UCB for logistic rewards.
k_mu = 1/4;
R_max = 1;
c_mu = 1/(2+exp(-xmax*max_norm_b)+exp(xmax*max_norm_b));

for t=1:T
    x = contexts(t,:)';

    %------ First, choose which arm we should pull in this round
    if(t == d*k+1)
        for i=1:k
            obs_filt = find((pull_ind(:,i)==1));
            Vt((i-1)*d+1:i*d, (i-1)*d+1:i*d) =
 contexts(obs_filt,:)'*contexts(obs_filt,:);
        end
    lambda_0 = min(eig(Vt));
    Vt_inverse = inv(Vt);
    end
    if (t>d*k)
        kappa = sqrt(3+2*log(1+(2*xmax^2)/lambda_0));
        rho_t = (2 * k_mu * kappa * R_max) / (c_mu) * ...
            sqrt(2 * k * d * log(t) * log(2 * k * d * T / delta));

        optimism_amount = zeros(k,1);
        for i=1:k
         optimism_amount(i) = x' * Vt_inverse(((i-1) * d + 1):(i *
 d), ...
             ((i-1) * d + 1):(i * d))*x;
        end
        optimistic_reward = exp(betahat * x) ./ (1 + exp(betahat * x))
 + ...
            rho_t * sqrt(optimism_amount);

        [~, imax] = max(optimistic_reward);
        arm_pulled = imax;
    else
        arm_pulled = mod(t, k)+1;
    end


    pull_ind(t, arm_pulled)=1;

    %------ Second, calculate the regret
```

```
    bx = b*x;
    [largest_inner_product, ~] = max(bx);
    bestreward = exp(largest_inner_product) / ...
        (1 + exp(largest_inner_product));
    ourreward = exp(bx(arm_pulled)) / (1 + exp(bx(arm_pulled)));


    if (t==1)
        regret(t) = bestreward - ourreward;
    else
        regret(t) = regret(t-1) + bestreward - ourreward;
    end


    % First updating XtopX inverse
    if(t >= k*d+1)
        ut = zeros(k*d,1);
        ut( ((arm_pulled-1)*d+1):(arm_pulled*d) ) = x;

        Vt_inverse_ut = Vt_inverse * ut;
        Vt_inverse = Vt_inverse - (Vt_inverse_ut *
 Vt_inverse_ut') / ...
            (1 + ut' * Vt_inverse_ut);


    % Update estimates
        obs_filt = find((pull_ind(:,arm_pulled)==1));
        lsX = contexts(obs_filt, :); % Design matrix.
        lsY = reward_vector(obs_filt, arm_pulled);
        if (size(lsX,1)>=d && rank(lsX)>=d)
            mdl = fitglm(lsX, lsY,'linear', 'Distribution', ...
                'binomial', 'Intercept', false);
            betahat(arm_pulled, :) = mdl.Coefficients.Estimate';
        end
    end

    if (verbose==1)
        if (mod(t,500)==0)
            fprintf('GLM-UCB: t=%d, Error in estimation = %f\n',
 t, ...
                norm(b-betahat,2))
        end
    end

end

fractions = mean(pull_ind);  %fraction of times each arm is pulled
if(verbose == 1)
    fprintf('GLM-UCB: Error in estimation = %f\n', norm(b-betahat,2));
    fprintf('GLM-UCB: Fraction of pulls = %f\n', fractions);
    fprintf('GLM-UCB: Regret Occured = %f\n',regret(end));
end

end
```

Published with MATLAB® R2015a