Table of Contents

rankoneupdate.m

```
% Calculates the solution of min_beta \|X*beta - y\|_2 using normal
% equations for least squares and Sherman-Morrison rank one update
 formula.
%
% Finding the least squares solution is equivalent to solving equation
% (X^\top*X)*beta = X^\top*y or equivalently
% beta^* = (X^\top*X)^{-1}*(X^\top*y). Letting beta_previous denote
 the
% least squares solution to \|X_previous*beta - y_previous\|_2, we can
 use
% Sherman-Morrison formula to find x_current, the least squares
 solution
% to min \|X*beta - y\|_2 where X = [X_previous, x_observed^\top] and
% y = [y_previous, y_observed].
%
% Other than calculating the least squares solution, the function also
% keeps track of the inverse covariance matrix, which is useful for a
 quick
% rank one update implementation.
%
```

Inputs:

```
XtopX_previous_inverse: Inverse of matrix X_previous^\top*X_previous,
    a d*d matrix.
beta_previous: Least squares solution given by min_beta
    \|X_previous*beta-y_previous\|_2, a d*1 vector.
x_observed: New observed vector of covariates, a d*1 vector.
y_observed: New observed reward.
```

Outputs:

```
XtopX_inverse: Updated inverse covariance matrix.
beta: Updated least square solution given by min_beta \|X*beta-y\|_2.
```

Example:

XtopX_previous_inverse = [1 0; 0 1]; beta_previous = [0; 0]; x = [1; 2]; y = 2; [XtopX_inverse, beta] = rankoneupdate(XtopX_previous_inverse, beta_previous, x, y)

Code:

```
function [XtopX_inverse, beta] =
 rankoneupdate(XtopX_previous_inverse, ...
                                               beta_previous, x, y)
d=size(XtopX_previous_inverse,1);
% Change in inverse covariance matrix using Sherman-Morrison formula.
XtopX_inverse_change = (XtopX_previous_inverse * (x * x') * ...
    XtopX_previous_inverse) / (1 + x'* XtopX_previous_inverse * x);
XtopX_inverse = XtopX_previous_inverse - XtopX_inverse_change;
% Update least squares after simplifying Sherman-Morrison formula.
beta = (eye(d) - (XtopX_previous_inverse * (x * x') ...
    / (1 + x' * XtopX_previous_inverse * x))) * beta_previous ...
    + y * XtopX_inverse * x;
end
```

Published with MATLAB® R2015a