CS 395T	Scalable Machine Learning	Fall 2014

Solutions to Homework 1

Lecturer: Inderjit Dhillon Date Due: Sep 15, 2014
Keywords: *Linear Algebra, Regression*

1. (a)
```
function [U,M,cvRMSE] = ALS(k, maxiter, R, lambda, P)
% matrix factorization via alternating minimization
%
% [U, M] = ALS(k, maxiter, train, lambda, P)
%
% INPUT:
%   k       - number of factors in the matrix factorization
%   maxiter - maximum number of iterations
%   R       - matrix to be factorized
%   lambda  - regularization term
%   P - structure variable:
%   P.probe, P.nnzs: data, indices from test set
%
% OUTPUT:
%   U  - k by nr matrix
%   M  - k by nc matrix
%   R is approximated by U'*M

[nr,nc] = size(R);
[Ix,Jx,xx] = find(R);
nnzs = find(R);
[Iy,Jy,yy] = find(R');

cc = histc(Jx,1:nc); % number of nonzeros in each column
rc = histc(Jy,1:nr);

% randomly initialize the matrix U and M
U = rand(k,nr);
M = rand(k,nc);

obj = zeros(maxiter,1);
relerr = 1e-6;
for t = 1:maxiter
    fprintf('iter (%d):\n',t);
    fprintf('  minimize M while fixing U ...');
    s = cputime;
    j = 1;
    for i = 1:nc
```

```
        if cc(i)>0
            subU = U(:,Ix(j:j-1+cc(i)));
            M(:,i) = (lambda*eye(k)+subU*subU')\(subU*xx(j:j-1+cc(i)));
            j = j+cc(i);
        else
            M(:,i) = zeros(k,1);
        end
    end
    fprintf(' %.2f seconds\n',cputime-s);
    fprintf('  minimize U while fixing M ...');
    s = cputime;
    j = 1;
    for i = 1:nr
        if rc(i)>0
            subM = M(:,Iy(j:j-1+rc(i)));
            U(:,i) = (lambda*eye(k)+subM*subM')\(subM*yy(j:j-1+rc(i)));
                        j = j+rc(i);
        else
            U(:,i) = zeros(k,1);
        end
    end
    fprintf(' %.2f seconds\n',cputime-s);
    Pred = U'*M;
    res = sum((xx - Pred(nnzs)).^2);
    obj(t) = .5*(res+lambda*(norm(U,'fro')^2+norm(M,'fro')^2));
    train = sqrt(res/length(xx));
    probe = sqrt(sum((P.probe-Pred(P.nnzs)).^2)/length(P.probe));
    fprintf('obj=%.4f rmse(train)=%.4f rmse(probe)=%.4f\n',obj(t),train,probe);
    if t > 1
        if ((obj(t-1)-obj(t))/obj(t-1) < relerr)
            break;
        end
    end
end
cvRMSE = probe;
end
```

(b) Regularization is key to performing the regression here. Without regularization, the matrix U (or M) is ill-conditioned — 'backslash' operator gives a warning to this effect, and the computed U or M is not useful.

(c) On the `small` dataset, $\lambda = 1$ is optimal. The corresponding test RMSE, with $k = 10$ and using 10 iterations is 1.0951.

(d) On the `medium` dataset, $\lambda = 1$ is optimal again. The corresponding test RMSE, with $k = 10$ and using 10 iterations is 0.8704. On the `large` dataset, with $\lambda = 1$, $k = 10$ and using 10 iterations, the test RMSE is 1.595.

2. (a) The results are as follows.

λ	training	test
0.1	0.1429	0.09285
1	0.1324	0.07751
10	0.1764	0.1557

(b) The results are as follows.

λ	training	test
0.1	0.1430	0.09269
1	0.1318	0.07103
10	0.2007	0.1664

(c) The results are as follows ($k = 10$, 10 ALS iterations, $\lambda = 1$).

		runtime (s)	test RMSE
`small`	`backslash`	5	1.0951
	`cd_ridge`	52	1.1628
`medium`	`backslash`	22	0.8704
	`cd_ridge`	292	0.8741
`large`	`backslash`	221	1.595
	`cd_ridge`	4529	1.6060

Source codes.

```
function [w] = cd_ridge(y, X, lambda)
  n = size(X,1);
  d = size(X,2);
  w = zeros(d,1);
  r = -y;
  h = sum(X.^2,1);

  for iter=1:20
    for j=1:d
      delta = -(r'*X(:,j)+lambda*w(j))/(lambda+h(j));
      w(j) = w(j) + delta;
      r = r + delta*X(:,j);
    end
    fprintf('iter %g obj %g\n', iter, 0.5*norm(X*w-y)^2+0.5*lambda*norm(w)^2);
  end

function [w] = cd_lasso(y, X, lambda)
  n = size(X,1);
  d = size(X,2);
  w = zeros(d,1);
  r = -y;
  h = sum(X.^2,1);

  for iter=1:20
    for j=1:d
      a = h(j);
      if (a==0)
        w(j) = 0;
      end
```

```
    b = -(r'*X(:,j)-h(j)*w(j));
    wnew = sign(b)*max(abs(b)-lambda,0)/a;
    delta = wnew-w(j);
    w(j) = w(j) + delta;
    r = r + delta*X(:,j);
  end
  fprintf('iter %g obj %g\n', iter, 0.5*norm(X*w-y)^2+lambda*sum(abs(w)));
end
```

Derivation for the Lasso coordinate descent update rule: The Lasso problem:

$$\operatorname*{argmin}_{\boldsymbol{w}} \frac{1}{2}\|X\boldsymbol{w}-\boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{w}\|_1 \equiv J(\boldsymbol{w}).$$

The one variable subproblem can be written as

$$\begin{aligned}
g(\delta) &= J(\boldsymbol{w}+\delta\boldsymbol{e}_i) \\
&= \frac{1}{2}\sum_{i=1}^{n}\left(\boldsymbol{x}_i^T\boldsymbol{w} - y_i + X_{ij}\delta\right)^2 + \lambda|w_j+\delta| + \sum_{i\neq j}\lambda|w_j| \\
&= \frac{1}{2}\sum_{i=1}^{n}\left(2(\boldsymbol{x}_i^T\boldsymbol{w} - y_i)X_{ij}\delta + X_{ij}^2\delta^2\right) + \lambda|w_j+\delta| + \text{ const} \\
&= \frac{\bar{\boldsymbol{x}}_j^T\bar{\boldsymbol{x}}_j}{2}\delta^2 + \boldsymbol{r}^T\bar{\boldsymbol{x}}_j\delta + \lambda|w_j+\delta| + \text{ const},
\end{aligned}$$

where $r_i = \boldsymbol{x}_i^T\boldsymbol{w} - y_i$. Now we do a variable transformation: let $h(d) = g(d - w_j)$. If d^* is the minimizer of $h(\cdot)$, δ^* is the minimizer of $g(\cdot)$, we can easily see that $d^* = w_j + \delta^*$. Also, we have

$$\begin{aligned}
h(d) &= \frac{\bar{\boldsymbol{x}}_j^T\bar{\boldsymbol{x}}_j}{2}(d-w_j)^2 + \boldsymbol{r}^T\bar{\boldsymbol{x}}_j(d-w_j) + \lambda|d| + \text{const} \\
&= \frac{\bar{\boldsymbol{x}}_j^T\bar{\boldsymbol{x}}_j}{2}d^2 + \left(\boldsymbol{r}^T\bar{\boldsymbol{x}}_j - \bar{\boldsymbol{x}}_j^T\bar{\boldsymbol{x}}_j w_j\right)d + \lambda|d| + \text{const}.
\end{aligned}$$

Let $a = \bar{\boldsymbol{x}}_j^T\bar{\boldsymbol{x}}_j$, $b = \boldsymbol{r}^T\bar{\boldsymbol{x}}_j - \bar{\boldsymbol{x}}_j^T\bar{\boldsymbol{x}}_j w_j$. If $a = 0$, then obviously the optimal solution is $d^* = 0$. If $a \neq 0$, $h(\cdot)$ is strictly convex, so the minimizer d^* is unique. Using the hint, we consider three cases:

(a) If $d^* > 0$, then $ad^* + b + \lambda = 0$, which implies $-\frac{b+\lambda}{a} = d^* > 0$. And since $a > 0$ we have $b < -\lambda$.

(b) If $d^* < 0$, then $ad^* + b - \lambda = 0$, which implies $-\frac{b-\lambda}{a} = d^* < 0$. And since $a > 0$, we have $b > \lambda$.

(c) Otherwise d^*=0.

Therefore, we have

$$\begin{aligned}
d^* &= \operatorname{sign}(-b)\max(|b|-\lambda, 0)/a, \\
\delta^* &= \operatorname{sign}(-b)\max(|b|-\lambda, 0)/a - w_j.
\end{aligned}$$