My MATLAB code sample starts from the next page. There are 99 lines in total. It is the core part of my OnTac system. OnTac is an online parameter-inference and task assignment system for crowdsourcing, detailed information can be found in “OnTac: Online Task Assignment for Crowdsourcing” (IEEE ICC 2016). The sample implements the task assignment phase and the inference phase, which is done by an online EM algorithm. This sample represents only one iteration of task assignment and inference. Due to the limitations on line numbers, I did not include the loops part and the parameter initialization part, and deleted something less important.

```
%% task assignment
%%% go over last iterated alpha to check whether we should trust labeler i
a_cache = [a_cache abs(alphaI(i))];
sac = size(a_cache);
if sac >= (sizeofcache + 1)
    a_cache(1) = [];
    if norm(a_cache) <= sqrt(thres*thres*sizeofcache)
        break;
    end
end
%%% rank the probabilities according to the distance with PermC
pRi    = 1 ./ ( 1 + exp(-alphaI(i)*betaI) );
pPerm  = pRi';
pPerm1 = abs(pPerm - PermC);
pPerm2 = abs(pPerm - (1-PermC));
pPermt = [pPerm1;pPerm2];
[vecp,rankp] = sortrows(pPermt);
pc = pcR;
alphaII   = alphaI(1:i);
%%% Estimate the expectation
pI = 1 ./ ( 1 + exp(-alphaII*betaI) );
for j = 1:nbq
    for ii = 1:i
        if Res(ii,j)==1
            pc(1,j) = pc(1,j) * (1 - pI(ii,j));
            pc(2,j) = pc(2,j) * pI(ii,j);
        elseif Res(ii,j)==0
            pc(1,j) = pc(1,j) * pI(ii,j);
            pc(2,j) = pc(2,j) * (1 - pI(ii,j));
        end
    end
end
for j = 1:nbq
    for ii = 1:i
        if Res(ii,j)~=-1
            pcND(:,j) = pcN(:,j);
        end
    end
end
%%% if the posterior proba is higher than a threshold, we
%%% think we get enough information
for j = 1:nbq
    if pc(1,j) > ConfC || pc(2,j) > ConfC
        findj = find(qConf==j);
        sfj   = size(findj);
        if sfj(2) == 0
             qConf = [];
        end
    end
end
alphaR = alphaI;
betaR  = betaI;
dis = 1;
%%% Maximization
while dis >0.01
    alphaI = alphaR;
    betaI  = betaR;
    for ii = 1:nbl
        for j = 1:nbq
            if Res(ii,j) == 0
                  alphaR(ii) = alphaR(ii) - 0.001*(     pcN(1,j) * ( -betaI(j) /
( exp(alphaI(ii)*betaI(j) ) +1)  )      +     pcN(2,j) *   ( -betaI(j) /
( exp(alphaI(ii)*betaI(j) ) +1)  + betaI(j) )          );
```

```
                betaR(j)  = betaR(j)  - 0.001*(     pcN(1,j) * (-alphaI(ii) /
( exp(alphaI(ii)*betaI(j) ) +1)  )      +     pcN(2,j) *   ( -alphaI(ii)/
( exp(alphaI(ii)*betaI(j) ) +1)  + alphaI(ii))          );
          elseif Res(ii,j) == 1
                alphaR(ii) = alphaR(ii) - 0.001*(     pcN(2,j) * ( -betaI(j) /
( exp(alphaI(ii)*betaI(j) ) +1)  )      +     pcN(1,j) *   ( -betaI(j) /
( exp(alphaI(ii)*betaI(j) ) +1)  + betaI(j) )          );
                betaR(j)  = betaR(j)  - 0.001*(     pcN(2,j) * (-alphaI(ii) /
( exp(alphaI(ii)*betaI(j) ) +1)  )      +     pcN(1,j) *   ( -alphaI(ii)/
( exp(alphaI(ii)*betaI(j) ) +1)  + alphaI(ii))          );
          end
       end
    end
    %%%% dis and disEM should be chosen between 1:i and i
    dis = (norm(alphaR-alphaI) + norm(betaR-betaI));
end
k = k+1;
%%% k+1 and i+1 because k and i are initially set to be 1
yita1 = (k+1)^(-par);
yita2 = (i+1)^(-par2);
if qcount == nbb
    alphaRP = alphaR;
else
    alphaRP = (1-yita1)*alphaRP + yita1*alphaR;
end
for j = vecRk
    if j ~=0
        if lastP(rk) == -1
            betaRP = betaR;
        else
        %%% should be studied. sometimes yita1 is better than yita2
            betaRP = (1-yita1)*betaRP + yita1*betaR;
        end
    end
end
```