Homework 4 - Model answer

1. Implement a nearest-neighbor classifier

Here is the function as Matlab, assuming a nCond x nVoxel x nRuns data matrix:

```
function acc=nn_classifier(data);
    nPart = size(data,3);  % Partitions are the 3 dimension
    nCond = size (data,1); % Number of conditons
    part = [1:nPart];
    for n=1:nPart
        trainIndx = find(part~=n);
        testIndx =  find(part==n);
        Mu_hat = mean(data(:,:,trainIndx),3); % Calculate the training means
        % Now classify
        for c=1:nCond
            x = data(c,:,testIndx);  % This is the test pattern
            dist=x*x'-2*Mu_hat*x'+sum(Mu_hat.^2,2);
            [~,k(c,n)]=min(dist);  % Record the classification
        end;
    end;
    % Caluclate the % correct
    correct=bsxfun(@eq,k,[1:5]');
    acc = sum(correct(:))/numel(correct(:));
```

Classification accuracy for the left (contralateral) hand: 0.675
Classification accuracy for the right (ipsilateral) hand: 0.400

2. Get confidence intervals from a randomization test

See case '1_classify' in homework4.m

Classification accuracy for the left (contralateral) hand: 0.675, $p<0.001$
Classification accuracy for the right (ipsilateral) hand: 0.400, $p=0.004$

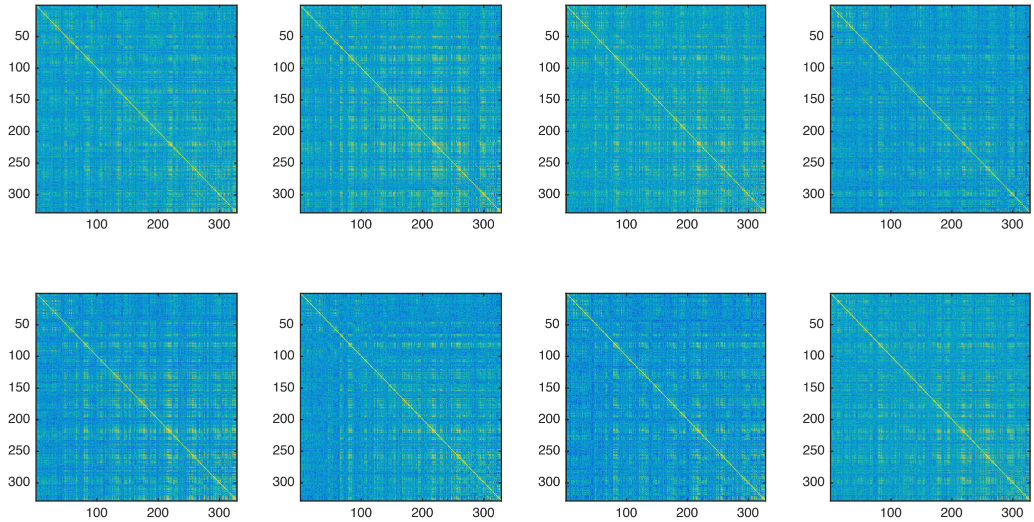
Both sides can be classified significantly better than chance. This is surprising, as the right motor cortex shows activity for the left hand, but suppression below baseline for the right hand!

Advanced question: Why can we not approximate the distribution of number of correct classifications as a binomial distribution?

Binomial distribution assumes independence across cases - this is not given, as training set for one classification becomes test set for another. That means that the classification accuracy in one crossvalidation fold is not independent from the classification in another crossvalidation fold. The violation is worse for less runs.

3. Prewhiten the data spatially using the residuals from 1-level regression

Figure 1 shows visually the spatial correlation matrix for the 8 runs. I show here the correlation, as each voxel is scaled to 1, such that the structure becomes more apparent than when plotting the covariance matrix. The matrices are highly similar to each other.



The whitening is implemented in the case ‘3_prewhiten’ in the attached function.

The repeated classification yields higher classification accuracies and therefore also higher p-values:

Classification accuracy for the left (contralateral) hand: 0.975, $p < 0.001$
Classification accuracy for the right (ipsilateral) hand: 0.450, $p < 0.001$

```
function varargout = homework4(what,varargin)
% Example matlab script to solve homework4
%
%
run=[]; % Run is a variable
switch(what)
    case '1_classify'
        handIndx=[1:5];   % For Left hand
        handIndx=[6:10];  % For Right Hand
        nRuns = 8;

        load dataset_4.mat;
        B=[];
        for r=1:nRuns
            X=[Xtask(:,:,r) Xhpf(:,:,r) Xintercept(:,:,r)];
            B(:,:,r)=pinv(X)*Y(run==r,:);
        end;

        B=B(handIndx,:,:);
        acc = nn_classifier(B);

        % Now randomly shufffle (Question 2)
        for i=1:1000
            for n=1:nRuns
                A(:,:,n)=B(randperm(5),:,n);
            end;
            sampacc(i)=nn_classifier(A);
        end;
        p=sum(sampacc>=acc)/numel(sampacc);
        fprintf('Classification accuracy: %2.3f, p=%2.6f\n',acc,p);
        varargout={acc,p};
```

```
    case '3_prewhiten'
         handIndx=[1:5];   % For Left hand
        handIndx=[6:10];  % For Right Hand
        nRuns =8;

        load dataset_4.mat;
        B=[];
        % Estimate Betas and residuals - plot Sigma estimates
        for r=1:nRuns
            N=size(Xhpf,1);
            R = eye(N)- Xhpf(:,:,r)*inv(Xhpf(:,:,r)'*Xhpf(:,:,r))*Xhpf(:,:,r)';

            X=[Xtask(:,:,r) Xintercept(:,:,r)];
            B(:,:,r)=pinv(R*X)*R*Y(run==r,:);
            Res(:,:,r)=R*Y(run==r,:)-R*X*B(:,:,r);
            Sig(:,:,r) = Res(:,:,r)'*Res(:,:,r);
            subplot(2,4,r);
            imagesc(corrcov(Sig(:,:,r)));
        end;

        % Average and regularize
        Sigma = mean(Sig,3);
        Sigma = 0.1*diag(diag(Sigma)) + 0.9*Sigma; % Shrinkage towards diagnoal
matrix

        % Prewhiten the data
        for r=1:nRuns
            B(:,:,r)=B(:,:,r)*Sigma^(-0.5);        % There are faster ways of
doing this
        end;


        % Redo the leave-one-out normalisation
        B=B(handIndx,:,:);
        acc = nn_classifier(B);
        % Now randomly shufffle
        for i=1:1000
            for n=1:nRuns
                A(:,:,n)=B(randperm(5),:,n);
            end;
            sampacc(i)=nn_classifier(A);
        end;
        p=sum(sampacc>=acc)/numel(sampacc);
        fprintf('Classification accuracy: %2.3f, p=%2.6f\n',acc,p);
        varargout={acc,p};
end
```