Classification With Randomer Forests

Table of Contents

Randomer forest is a sparse oblique decision forest method for classification.

Here we demonstrate how to train muliple models over a grid of parameter values, select the optimal model based on out of bag estimates, and make predictions on a test set using the selected model.

Load Iris Dataset

```
close all
clear
clc

load fisheriris
X = meas;
Y = cellstr(num2str(grp2idx(species))); %Convert to strings of numbers
classes = unique(Y);
[ntrain,p] = size(X);
```

Train Randomer Forest

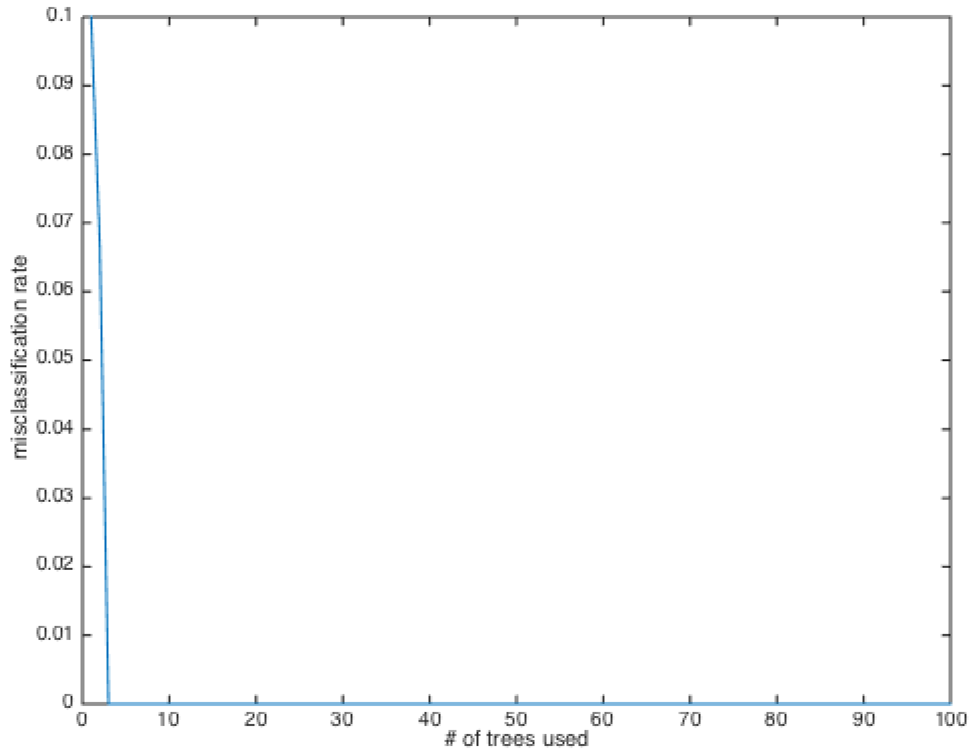
```
% Use 100 trees, very sparse Rademacher matrix (default) for random projections,
% sample d candidate projections at each split node, specify stratified
% bootstrap sampling, and connect to two parallel workers

Params.nTrees = 100;
Params.ForestMethod = 'rerf';
Params.RandomMatrix = 'binary';
Params.d = [1:p ceil(p.^[1.5 2])];  % one model will be fit for each value of d
Params.NWorkers = 2;
Params.Stratified = true;

% Partition into train and test set

trainIdx = [1:40 51:90 101:140];
testIdx = setdiff(1:150,trainIdx);
Xtrain = X(trainIdx,:);
Ytrain = Y(trainIdx);
Xtest = X(testIdx,:);
Ytest = Y(testIdx);

% Train a RerF for each value of Params.d

Forest = RerF_train(Xtrain,Ytrain,Params);
```

```
% Compute out-of-bag-error and AUC for each forest model corresponding to
% the values of Params.d. The best model will be chosen as the one having
% the lowest OOB error. If multiple models have the lowest OOB error, then
% the OOB AUC is used to attempt to break ties. If still multiple models
% are equally good, then the one using the largest value of Params.d is
% selected as the best one

for i = 1:length(Forest)

    % First compute scores (estimates of posterior probabilities of being
    % in each class). The 'last' flag indicates that we only want to
    % compute one set of scores using the whole forest.

    Scores = rerf_oob_classprob(Forest{i},...
        Xtrain,'last');

    % Next make OOB predictions by choosing the class with the largest
    % score as the predicted class

    Predictions = predict_class(Scores,Forest{i}.classname);

    % Compute OOB error from predictions

    OOBError(i) = misclassification_rate(Predictions,Ytrain,false);

    % Since there are more than two classes, OOB AUC is computed by
    % one-hot-encoding Ytrain using the function binarize_labels()

    if size(Scores,2) > 2
        Yb = binarize_labels(Ytrain,Forest{i}.classname);
        [~,~,~,OOBAUC(i)] = perfcurve(Yb(:),Scores(:),'1');
    else
        [~,~,~,OOBAUC(i)] = perfcurve(Ytrain,Scores(:,2),'1');
    end
end

% Select best model according to OOB errors and OOB AUCs

BestIdx = hp_optimize(OOBError,OOBAUC);
if length(BestIdx)>1
    BestIdx = BestIdx(end);
end
```

Make predictions on test set

```
% Compute scores on test set. The 'every' flag, as opposed to the 'last'
% flag, specifies that we want to compute a set of scores for every number
% of trees. Use 'every' only when you want to see how the error converges
% as a function of the number of trees used in the forest

Scores = rerf_classprob(Forest{BestIdx},Xtest,'every');
```

```
for t = 1:Forest{BestIdx}.nTrees

    % Make predictions using trees 1:t

    Predictions = predict_class(Scores(:,:,t),Forest{BestIdx}.classname);

    % Compute error rate on test set using trees 1:t

    TestError(t) = misclassification_rate(Predictions,Ytest,false);
end


plot(1:Params.nTrees,TestError)
xlabel('# of trees used')
ylabel('misclassification rate')
```



Published with MATLAB® R2014b