Classification of images from the web

Basic example of downloading images from the web and doing classifications

Wednesday

Anton Antonov
Digital Humanities at Oxford Summer School 2017
July 2017

Mission statement

With this notebook we show how to obtain, prepare, and do classification experiments with a curated database of images from the Web.

Secondary goal is to compare different classifiers.

The section "NNMF basis" follows the Markdown document "Handwritten digits classification by matrix factorization" from MathematicaVsR at GitHub.

UCI Li photograph

Actually this page is better. Also see this page.

```
dirName = "/Users/yaman/Downloads/Anton-notebooks-DHOxSS-2017/li_photograph/";

annotationText = ReadList[dirName <> "annotation.txt", String];

ColumnForm[annotationText[[1 ;; 4]] (*ilk dort row*)
```

```
0 1/10000.jpg      flower
1 1/10001.jpg      maple leaves
2 1/10002.jpg      flower
3 1/10003.jpg      flower
```

```
annotations =
  Map[StringCases[#, n : (DigitCharacter ..) ~~ (WhitespaceCharacter ..) ~~
        name : (Except[WhitespaceCharacter] ..) ~~ (WhitespaceCharacter ..) ~~
        class : (___) :> {n, name, class}][[1]] &, annotationText];
Dimensions[annotations]
```

```
{2360, 3}
```

```
{2360, 3}
```

```
annotations
```

```
{2360, 3}
```

```
{{0, 1/10000.jpg, flower}, {1, 1/10001.jpg, maple leaves},
 {2, 1/10002.jpg, flower}, {3, 1/10003.jpg, flower},
 {4, 1/10004.jpg, flower}, {5, 1/10005.jpg, flower}, ··· 2348 ···,
 {2354, 5/50089.jpg, coast}, {2355, 5/50090.jpg, sheep},
 {2356, 5/50091.jpg, sheep}, {2357, 5/50092.jpg, sheep},
 {2358, 5/50093.jpg, sheep}, {2359, 5/50094.jpg, sheep}}
```

large output | show less | show more | show all | set size limit...

```
RecordsSummary[annotations[[All, 3]][[1]]
```

··· Symbol: Symbol called with 0 arguments; 1 argument is expected.

```
Symbol[]
```

```
imageSubDirName = dirName <> "image.cd/"
```

```
/Users/yaman/Downloads/Anton-notebooks-DHOxSS-2017/li_photograph/image.cd/
```

```
"/Users/yaman/Downloads/Anton-notebooks-DHOxSS-2017/li_photograph/image.cd/"
```

```
AbsoluteTiming[
 images = Map[Import[imageSubDirName <> #[[2]] &, annotations];
 (*comcreateing the images. takes the second element and glues them*)
]
```

```
{253.735, Null}
```

```
AbsoluteTiming[
 images = ConformImages[images];
 (*fotolari sekilleyebiliyormusuz bu functionla*)
]
```

```
{5.48999, Null}
```

```
Tally[ImageDimensions /@ images]
```

```
{{{384, 256}, 2360}}
```

```
{{{384, 256}, 2360}}
```

```
{{{384, 256}, 2360}}
```

```
Length[images]
```

```
{{{384, 256}, 2360}}
```

```
2360
```

```
data =
  Thread[
   images →
    Map[Which[
       # == "flower" || # == "water plant, flower", "flower",
       # == "firework", #,
       True, "other"
      ] &, annotations[[All, 3]]]
  ]; (*BURDA HATA VERIYOR! CRASH YAPIYOR*)
```

```
RandomSample[data, 6]
```

RandomSample: The set of items to sample from, data, should be a non–empty list or a rule weights –> choices.

```
RandomSample[data, 6]
```

```
trainingInds = Flatten@Map[(pos = Flatten[Position[data[[All, 2]], #]];
      RandomSample[pos, Round[0.7 Length[pos]]]) &, Union[data[[All, 2]]]];
testInds = Complement[Range[Length[data]], trainingInds];
```

```
Length /@ {trainingInds, testInds}
```

```
{1653, 707}
```

```
Tally[data[[All, -1]]]
```

```
{{flower, 385}, {other, 1907}, {firework, 68}}
```

```
If[True,
 data[[All, 1]] = ConformImages[data[[All, 1]], {56, 56}, ColorSpace → "Grayscale"];
]
```

With a "NeuralNetwork" classifier

```
AbsoluteTiming[
 flowerFunc = Classify[data[[trainingInds]], Method -> "NeuralNetwork"]
]
```

```
{5.6196, ClassifierFunction[ Input type: Image
                             Classes: firework, flower, other ]}
```

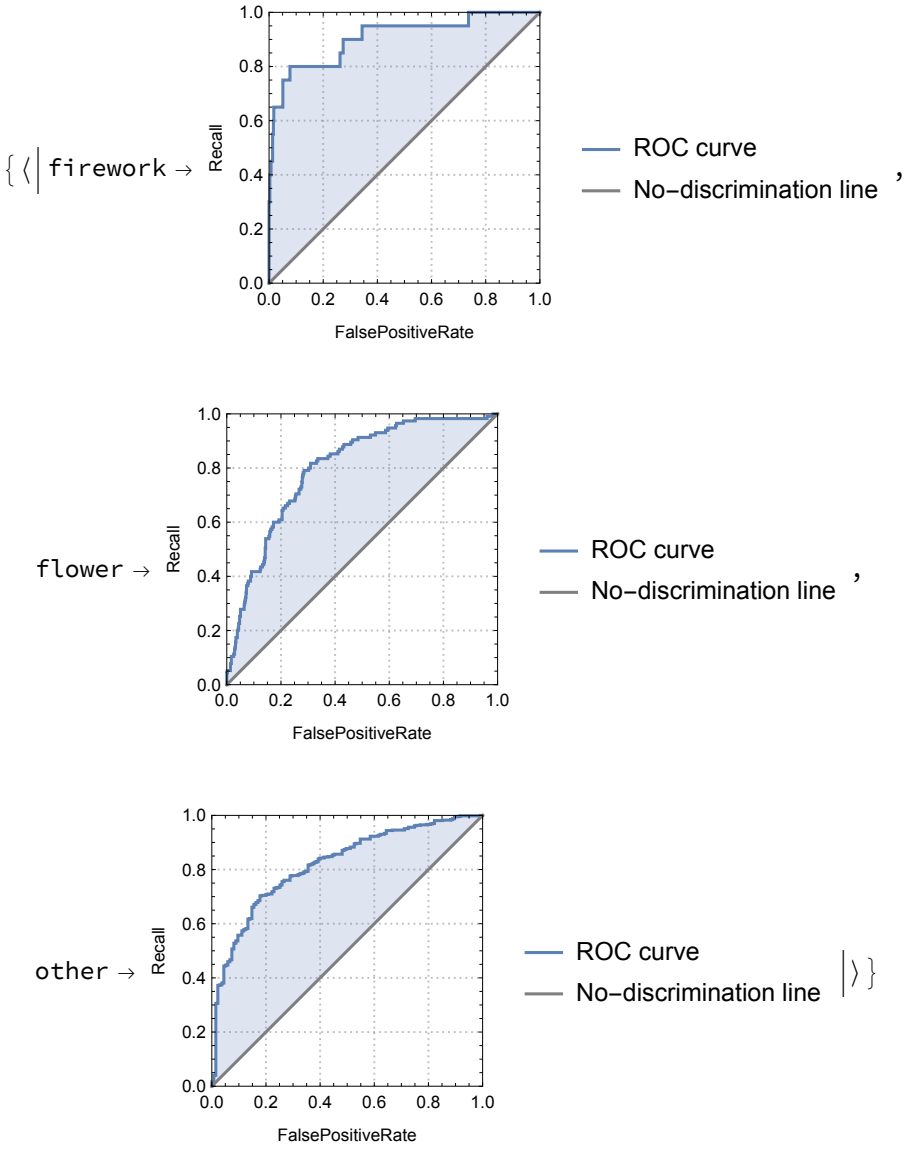
```
AbsoluteTiming[
 cm = ClassifierMeasurements[flowerFunc, data[[testInds]]]
]
```

```
{0.114986, ClassifierMeasurementsObject[ Classifier: NeuralNetwork
                                         Number of test examples: 707 ]}
```

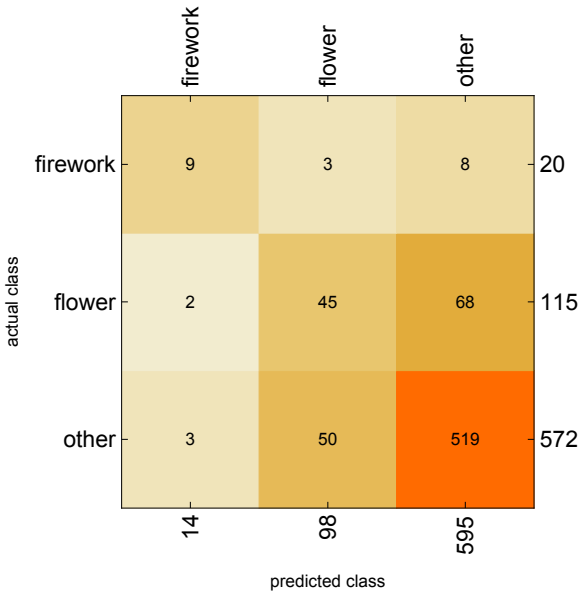
```
cm[{"Accuracy", "Precision", "Recall"}]
```

```
{0.810467, ⟨|firework → 0.642857, flower → 0.459184, other → 0.872269|⟩,
 ⟨|firework → 0.45, flower → 0.391304, other → 0.907343|⟩}
```

```
cm[{"ROCCurve"}]
```


{⟨|firework →
Recall
FalsePositiveRate
ROC curve
No−discrimination line
,
flower →
Recall
FalsePositiveRate
ROC curve
No−discrimination line
,
other →
Recall
FalsePositiveRate
ROC curve
No−discrimination line
|⟩}

```
cm["ConfusionMatrixPlot"]
```



With a “NearestNeighbors” classifier

```
AbsoluteTiming[
 flowerFunc = Classify[data[[trainingInds]], Method -> "NearestNeighbors"]
]
```

```
{5.20248, ClassifierFunction[ Input type: Image
Classes: firework, flower, other ]}
```

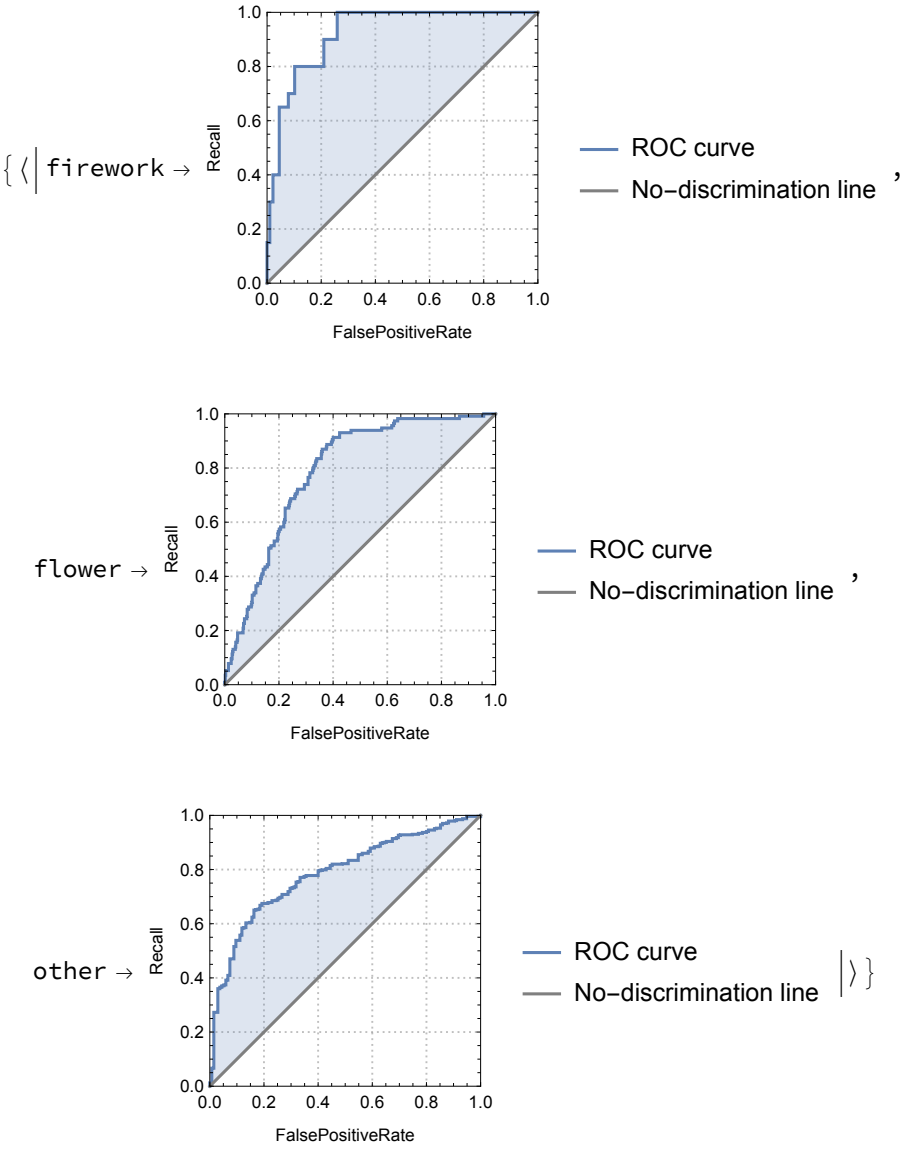
```
AbsoluteTiming[
 cm = ClassifierMeasurements[flowerFunc, data[[testInds]]]
]
```

```
{0.134498, ClassifierMeasurementsObject[ Classifier: NearestNeighbors
Number of test examples: 707 ]}
```

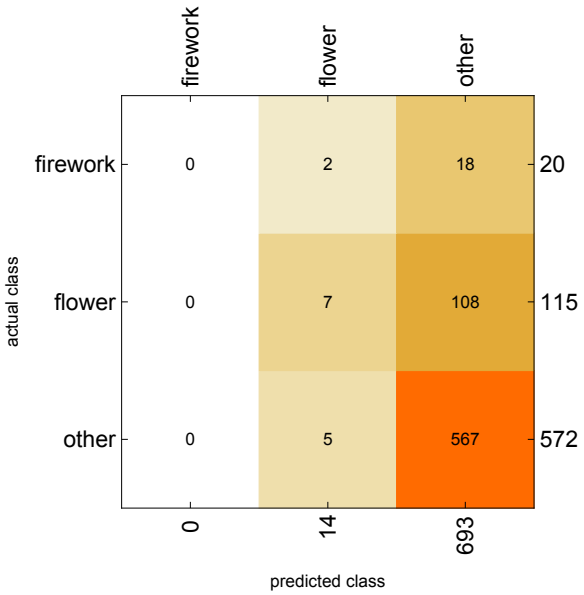
```
cm[{"Accuracy", "Precision", "Recall"}]
```

```
{0.811881, ⟨| firework → Indeterminate, flower → 0.5, other → 0.818182 |⟩,
 ⟨| firework → 0., flower → 0.0608696, other → 0.991259 |⟩}
```

```
cm[{"ROCCurve"}]
```


{⟨|firework →
Recall
FalsePositiveRate
ROC curve
No−discrimination line
,
flower →
Recall
FalsePositiveRate
ROC curve
No−discrimination line
,
other →
Recall
FalsePositiveRate
ROC curve
No−discrimination line
|⟩}

```
cm["ConfusionMatrixPlot"]
```



NNMF basis

Here we follow the Markdown document “Handwritten digits classification by matrix factorization” from MathematicaVsR at GitHub.

Getting data

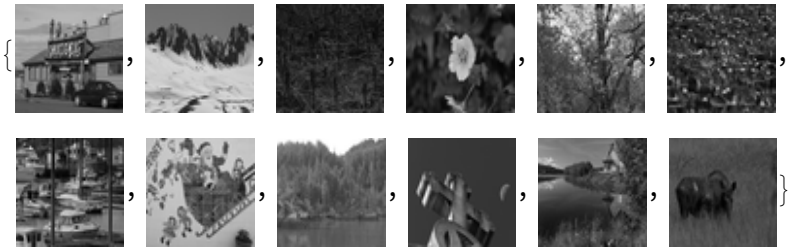
```
Reverse@Sort@Select[Tally[annotations[[All, 3]]], #[[2]] > 20 &]
```

```
{{water, sea otter, 41}, {water plant, flower, 57},
 {water plant, 41}, {water, 22}, {tree, 33}, {snow mountain, 41},
 {rock, water, 21}, {rock, tree, bald eagle, 21},
 {rock, bird, 41}, {plant, 42}, {leaves, 23}, {ice, water, 24},
 {iceberg, water, snow mountain, 21}, {grass land, animal, bear, 23},
 {glacier, snow, 37}, {glacier, 24}, {flower, 328}, {firework, 68},
 {boat, 22}, {bird, water, grass, 30}, {bird, water, 43}, {bird, 42}}
```

```
data =
  Thread[images → Map[Which[# == "flower" || # == "water plant, flower", "flower",
       # == "firework", #, True, "other"] &, annotations[[All, 3]]]];
```

```
trainImages =
  ConformImages[data[[trainingInds, 1]], {56, 56}, ColorSpace → "Grayscale"];
trainImagesLabels = data[[trainingInds, 2]];
```

```
RandomSample[trainImages, 12]
```



```
RandomSample[trainImagesLabels, 12]
```

```
{flower, other, other, other, flower,
 flower, other, other, other, flower, other, flower}
```

```
testImages = ConformImages[data[[testInds, 1]], {56, 56}, ColorSpace → "Grayscale"];
testImagesLabels = data[[testInds, 2]];
```

Linear vector space representation

```
trainImagesMat = N@SparseArray[Flatten@* ImageData /@ trainImages]
```

```
SparseArray[ Specified elements: 5180515
             Dimensions: {1653, 3136} ]
```

```
testImagesMat = N@SparseArray[Flatten@* ImageData /@ testImages]
```

```
SparseArray[ Specified elements: 2215256
             Dimensions: {707, 3136} ]
```

```
classLabels = Union[trainImagesLabels, testImagesLabels]
```

```
{firework, flower, other}
```

The matrix factorization

```
nBasisSize = 40;
AbsoluteTiming[
 nnmfRes = ParallelMap[Function[{cl}, Print[Style[cl, Bold, Red]];
    pos = Flatten@Position[trainImagesLabels, cl];
    tmat = trainImagesMat[[pos, All]];
    res = GDCLS[tmat, nBasisSize, PrecisionGoal → 4, "MaxSteps" → 20,
      "RegularizationParameter" → 0.1, "PrintProfilingInfo" → False];
    bres = RightNormalizeMatrixProduct @@ res;
    Join[bres,
     {(Norm /@ res[[2]]) / (Norm /@ bres[[2]]), PseudoInverse[bres[[2]]], tmat}]],
   classLabels, DistributedContexts → Automatic];
 nnmfRes = AssociationThread[classLabels → nnmfRes];]
```

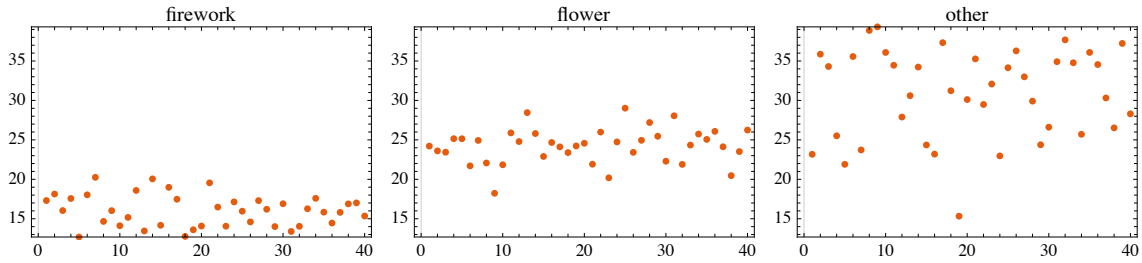
firework

flower

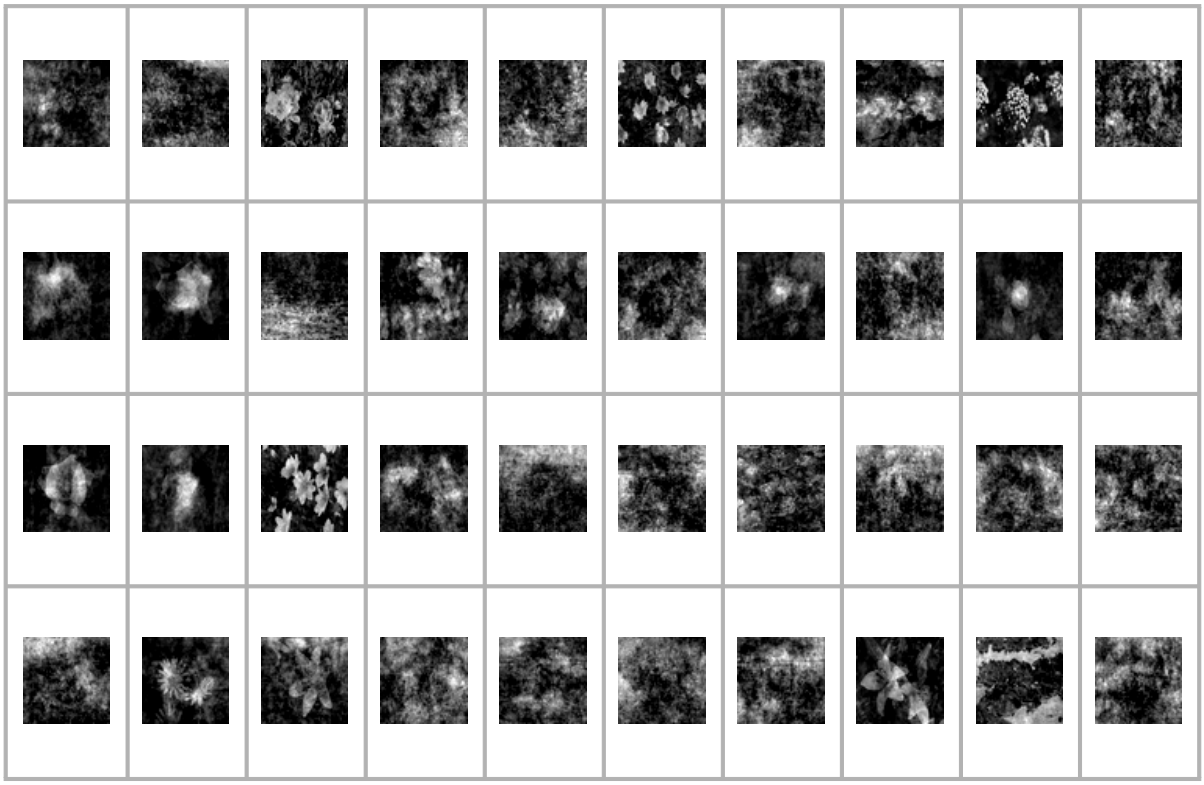
other

{170.249, Null}

```
Grid[ArrayReshape[#, {4, 3}, ""] &@Map[ListPlot[nnmfRes[#][[3]],
      PlotRange → MinMax[Flatten@Normal@Values[nnmfRes][[All, 3]]],
      PlotStyle → PointSize[0.02], PlotTheme → "Scientific",
      ImageSize → 190, PlotRange → All, PlotLabel → #] &, Keys[nnmfRes]]]
```




```
Magnify[#, 1.6] &@Grid[Partition[#, 10] &@
    Map[ImageAdjust@Image@Partition[#, 56] &, nnmfRes["flower"][[2]]],
  Dividers → All, FrameStyle → GrayLevel[0.7]]
```



Classification functions

```
Clear[NNMFClassifyImageVector]
Options[NNMFClassifyImageVector] =
  {"PositiveDifference" → False, "NumberOfNNs" → 4,
   "WeightedNNsAverage" → False, "RepresentationNorm" → False};
NNMFClassifyImageVector[factorizationRes_Association, vec_?VectorQ,
   opts : OptionsPattern[]] := Block[{residuals, invH, nW, nf, approxVec,
    scores, pos, rv, nnns = OptionValue["NumberOfNNs"], inds, ws},
   residuals = Map[(invH = factorizationRes[#][[4]];
       (*nW=(#/Norm[#])&/@factorizationRes[#][[1]];*)
       nf = Nearest[factorizationRes[#][[1]] →
          Range[Dimensions[factorizationRes[#][[1]]][[1]]]];
       If[TrueQ[OptionValue["RepresentationNorm"]], approxVec = vec.invH;
        CosineDistance[Flatten[factorizationRes[#][[1]][[nf[approxVec]]]],
         approxVec], (*ELSE*)inds = nf[vec.invH, OptionValue["NumberOfNNs"]];
        If[TrueQ[OptionValue["WeightedNNsAverage"]],
         ws = Map[Norm[vec - #] &, (factorizationRes[#][[5]])[[inds]]];
         approxVec = ws.(factorizationRes[#][[5]])[[inds]], (*ELSE*)
         approxVec = Total[(factorizationRes[#][[5]])[[inds]]]];
        rv = vec / Norm[vec] - approxVec / Norm[approxVec];
        If[TrueQ[OptionValue["PositiveDifference"]], rv = Clip[rv, {0, ∞}]];
        Norm[rv]]) &, Keys[factorizationRes]];
   {Keys[factorizationRes][[Position[residuals, Min[residuals]][[1, 1]]]],
    AssociationThread[Keys[factorizationRes] → residuals]}];
```

Classification

```
AbsoluteTiming[nnmfClResInv =
   ParallelMap[NNMFClassifyImageVector[nnmfRes, #, "RepresentationNorm" → False,
      "NumberOfNNs" → 30, "WeightedNNsAverage" → False] &, # & /@ testImagesMat];]
```

```
{5.78741, Null}
```

```
nnmfClResDT = Transpose[{testImagesLabels, nnmfClResInv[[All, 1]]}];
```

Total accuracy

```
N@Mean[(Equal @@@ nnmfClResDT) /. {True → 1, False → 0}]
```

```
0.746818
```

Precision per class

```
t = Map[Association@Flatten@{"Label" → #[[1, 1]], "NImages" → Length[#],
        "Precision" → N@Mean[(Equal @@@ #) /. {True → 1, False → 0}]} &,
    GatherBy[nnmfClResDT, First]];
t = SortBy[t, First];
Dataset[t]
```

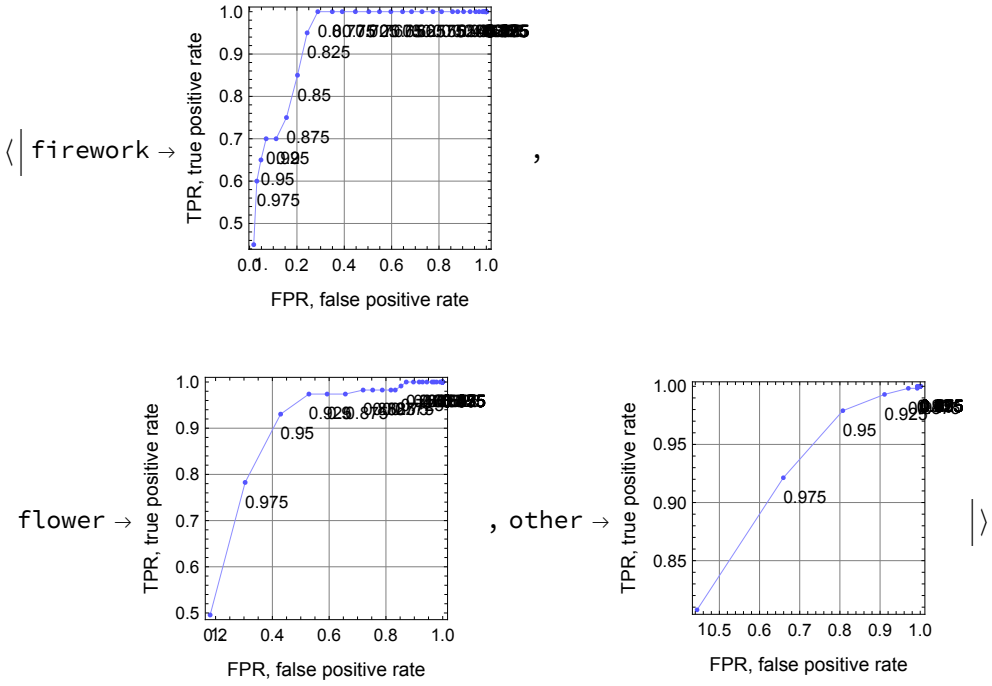
ROC curve

Fairly complicated to computer here!

```
thRange = Range[0, 1, 0.025];
```

```
aROCs =
  Association@
   Table[(
     nonTargetClass = "Non" <> targetClass;
     targetClass →
      Table[(
        mf = Join[AssociationThread[
            classLabels → Table[1, Length[classLabels]]], <|targetClass → th|>];
        clRes = Map[Merge[{mf, #}, Times @@ # &] &, nnmfClResInv[[All, 2]]];
        cres = Position[#, Min[#]][[1, 1, 1]] & /@ clRes;
        cres = If[# == targetClass, #, nonTargetClass] & /@ cres;
        ToROCAssociation[{targetClass, nonTargetClass}, Map[If[# == targetClass,
            #, nonTargetClass] &, testImagesLabels], cres]), {th, thRange}]),
     {targetClass, classLabels}
    ];
```

```
AssociationMap[
 ROCPlot[thRange, aROCs[#], "PlotJoined" → Automatic, GridLines → Automatic] &,
 Keys[aROCs]]
```



Getting artsy