Comment on the threshold being irrelevant as long as it's between 0.05 and 0.2, as BOTH N and U can change probabilities, hence the algorithm will converge anyway. Show that I've tried this!

Implementing S-EM:

---

1. Independent method for I-EM

2. DataSet to be used is the 20newsgroups. As the positive label, take those labeled as 1.

OK, I NEED A METHOD TO TELL WHICH PARTS TO USE AS POSITIVE AND UNLABELED, AND WANT IT TO RETURN THE REQUIRED INPUT TO IEM AND SEM.

This will make sampling and applying the algorithm much simpler.

Method created. We also have a method, qualityOfClassifier, that can asses the accuraccy, recall and f-score of our classification.

Now, testing for any positive and unlabeled set is pretty straightforward. The prepareData method can be used for preparing data for use with Bayesian sets and well, and we can easily compare their performance, albeit the fact that Bayesian sets is a ranking method, but we can compare how it's N top results compare to the N values that S-EM added to the cluster. Potentially, we can regard the posteriors in the S-EM as a continuous scoring metric for these values, and hence compare with some other metric?? Probably not.

[FeatureMatrix, trueLabels] = prepareData(doc,

1:100, 1000:16200, newsgroups);

>> Result = InitialEM(FeatureMatrix, 1:100, 101:

length(trueLabels));

iterationCount =

18

ans =

3930

>> [pr, rc, fsc] = entitySetExpQuality

(binarizePosteriors(Result), trueLabels);

>> pr

pr =

0.7882

>> rc

rc =

0.8324

>> fsc

fsc =

0.8097

This demonstrates the use of applying iEM and

assesing the quality of its performance.

Step 3: S-EM algorithm

Sample, do IEM, split into three sets, P, U, N.

N is least likely to contain positives. Just for

future statistical evaluation, the sEM function

returns these three sets. Just by ploting U and N,

we can see that N is significantly steeper, i.e.

contains almost no elements less than 4000, which

is the positive set. On the other hand, the

"purity" of the unlabeled set is very, very high -

more than 50% of the elements are actually

positives!!!

NB: Reshuffling the feature matrix before applying

iEM, (potentially) SEM, or even NB, is worthwhile

in saving time restoring the original indices

after sorting to take out Negative Set.

The main source of confusion is still the

"probabilistic label", and the convergence of the

classifier. If we were to relabel at each step

probabilistically, that might mean that we never

converge in labels - but we might converge in

posteriors? Answer: ok, so if we just assign

probabilistic labels, labels indeed do not

converge - logical. Now, about posteriors? This

actually seems to be converging (to 6500 given

4000, not unsensible, but seems to be quite slow,

100+ iterations, still didn't converge) therefore,

stick to my original >=0.5 binarization, which

ensures deterministic labeling and hence

convergence in both labels and posteriors. Also,

much better pr, rc, fsc.

Now, given P, N, U, we need to proceed with the

final EM steps of sEM.

TODO: Write documentation on this; Compare iEM to

sEM.

Sample execution and comparison:

>> [FeatureMatrix, trueLabels] = prepareData(doc,

1:100, 1000:16200, newsgroups);

Error using doc

Too many output arguments.

>> [FeatureMatrix, trueLabels] = prepareData(doc,

1:100, 1000:16200, newsgroups);

Error using doc

Too many output arguments.

>> doc = double(documents)';

>> [FeatureMatrix, trueLabels] = prepareData(doc,

1:100, 1000:16200, newsgroups);

>> [Result, P, U, N] = SpyEM(FeatureMatrix, 1:100,

101: length(trueLabels));

>> [pr, rc, fsc] = entitySetExpQuality

(binarizePosteriors(Result), trueLabels);

>> [pr, rc, fsc]

ans =

0.7882 0.8324 0.8097

>> Result = InitialEM(FeatureMatrix, 1:100, 101:

length(trueLabels));

>> [pr, rc, fsc] = entitySetExpQuality

(binarizePosteriors(Result), trueLabels);

>> [pr, rc, fsc]

ans =

0.7882 0.8324 0.8097

>> sum(trueLabels)

ans =

3706

Example of (small) difference of iEM and sEM:

>> [FeatureMatrix, trueLabels] = prepareData(doc,

1:200, 1000:16200, newsgroups);

>> [Result, P, U, N, iterations] = SpyEM

(FeatureMatrix, 1:200, 201: length(trueLabels),

0.15, trueLabels);

ans =

0.7912 0.8460 0.8177

>> Result = InitialEM(FeatureMatrix, 1:200,

201:length(trueLabels));

ans =

0.7896 0.8382 0.8132

for training with only 10 positives, we get the

same result! Exactly the same!

For a large training sample, and a large number of

entries to be added to the entity set, trying with

increasing the size of positive set w.r.t. hidden

positives:

>> [FeatureMatrix, trueLabels] = prepareData(doc,

1:2000, 2001:16200, newsgroups);

>> sum(trueLabels)

ans =

4605

>> [Result, P, U, N, iterations] = SpyEM

(FeatureMatrix, 1:200, 201: length(trueLabels),

0.15, trueLabels);

ans =

0.7914 0.8756 0.8313

>> [Result, P, U, N, iterations] = SpyEM

(FeatureMatrix, 1:2000, 2001: length(trueLabels),

0.15, trueLabels);

ans =

0.7508 0.9338 0.8324

>> [Result, P, U, N, iterations] = SpyEM

(FeatureMatrix, 1:4000, 4001: length(trueLabels),

0.15, trueLabels);

ans =

0.7457 0.9857 0.8490

Hence, increasing ratio of known positives to

unknows positives reduces precision, increases

recall, and (gently) improves the fscore.

Observe what results iEM obtains:

1:200 for positive:

ans =

0.7888 0.8734 0.8289

1:2000 for positive:

ans =

0.7455 0.9075 0.8185

1:4000 for positive:

ans =

0.7316 0.9164 0.8137

Therefore, we can see that sEM obtains an

improvement in any setting, but the more

substantial improvements in fscore come with an

increased number of positive training samples!

Concerning the negative threshold, t, we can see

that 0.05 and 0.15 get the EXACT same results,

just as the paper claims!

>> [Result, P, U, N, iterations] = SpyEM

(FeatureMatrix, 1:2000, 2001: length(trueLabels),

0.15, trueLabels);

after 17 iterations:

ans =

0.7508 0.9338 0.8324

>> [Result, P, U, N, iterations] = SpyEM

(FeatureMatrix, 1:2000, 2001: length(trueLabels),

0.05, trueLabels);

after 13 iterations:

ans =

0.7508 0.9338 0.8324

Therefore, exactly the same results - just a

different number of iterations needed to converge!

TODO TODO TODO:::: after calculating all of this,

and do qualEstimate, they still yield the same

result : GLITCH WHEN BINARIZING POSTERIORS OR

SOMETHING!!!!!!!!! What was happening was that we

weren't ensuring that the output of the algorithm

fixed positives as 1 when binarizing, hence not

employing the condition that the probabilities of

the entries in the positive set should all be set

to 1!

TODO TODO: Make a precision@N or some other metric

for assesing Bayesian sets

Notes on performance metrics: F-measure

Main article: F-score

The weighted harmonic mean of precision and

recall, the traditional F-measure or balanced F-

score is the fscore.

Implemented Precision@N, example:

>> [listIdx, listScore] = produceRankingDirect

(scoreList, 1:length(scoreList), 0);

>> precN = precisionAtN(listIdx, trueLabels, 5606)

precN = 0.7364

where 5606 is the number classified as positive by

SEM.

For the same example, sEM yields:

[pr, rc, fsc]

ans = 0.7455 0.9075 0.8185

or, for another example(same data, different P):

>> [Result, P, U, N, iterations] = SpyEM

(FeatureMatrix, 1:1000, 1001: length(trueLabels),

0.05, trueLabels);

>> [pr, rc, fsc] = entitySetExpQuality

(binarizePosteriors(Result), trueLabels);

>> [pr, rc, fsc]

ans =

0.7616 0.9136 0.8307

>> scoreList = BayesianSet(FeatureMatrix', 1:1000,

2);

>> [listIdx, listScore] = produceRankingDirect

(scoreList, 1:length(scoreList), 0);

>> sum(binarizePosteriors(Result))

ans =

5524

>> sum(trueLabels)

ans =

4605

>> precN = precisionAtN(listIdx, trueLabels, 5524)

precN =

0.7178

>> precN = precisionAtN(listIdx, trueLabels, 4605)

precN =

0.7789

We can see that in this case, on a ranking of the

first 5524 scores ( the number classified as

positive by sEM), Bayesian sets actually achieve a

better result! It is possible that the claim in

the paper is different from the actual

understanding of the problem and quality of

estimation and relies too much on the similarity

between posteriors and Bayesian sets scores... or not??

In general, potential rethink of evaluation metrics used, and potential of using different metrics for gaining new insight??

TODO:

1. Rocchio & SVM as another potential PU method.

2. Iterative Bayesian sets algorithm.

3. Read "Growing a list".

4. Read "when PU > binary classification - biased data sets as realistic data instances"