```
function [allScores, allFPs, allMisses, allMerges] = compareClustering2(cluGT, resGT, cluTest, resTest)
```

已知：

Ground Truth (GT)		Prediction (Test)	
真实的 spike peak times	每个 spike 对应的 neuron (共30个)	预测的 spike peak times	检测该 spike 所用的 optimized template (cluster)，共64个
resGT	cluGT	resTest	cluTest
131	23	130	34
173	30	172	24
412	4	411	33
532	20	531	15
534	2	534	7
613	6	611	35
618	3	617	64
648	10	647	3
777	25	776	5
900	16	899	26
987	15	986	48
1004	19	1003	9
1450	16	1449	26
1517	10	1516	3
1759	12	2058	4
⋮	⋮	⋮	⋮

每个优化得到的 cluster (template) 对应的 detected spikes

$nSp = (9022, 9355, 6467, \cdots)_{1\times 64}$

对于每一个真实的 neuron：

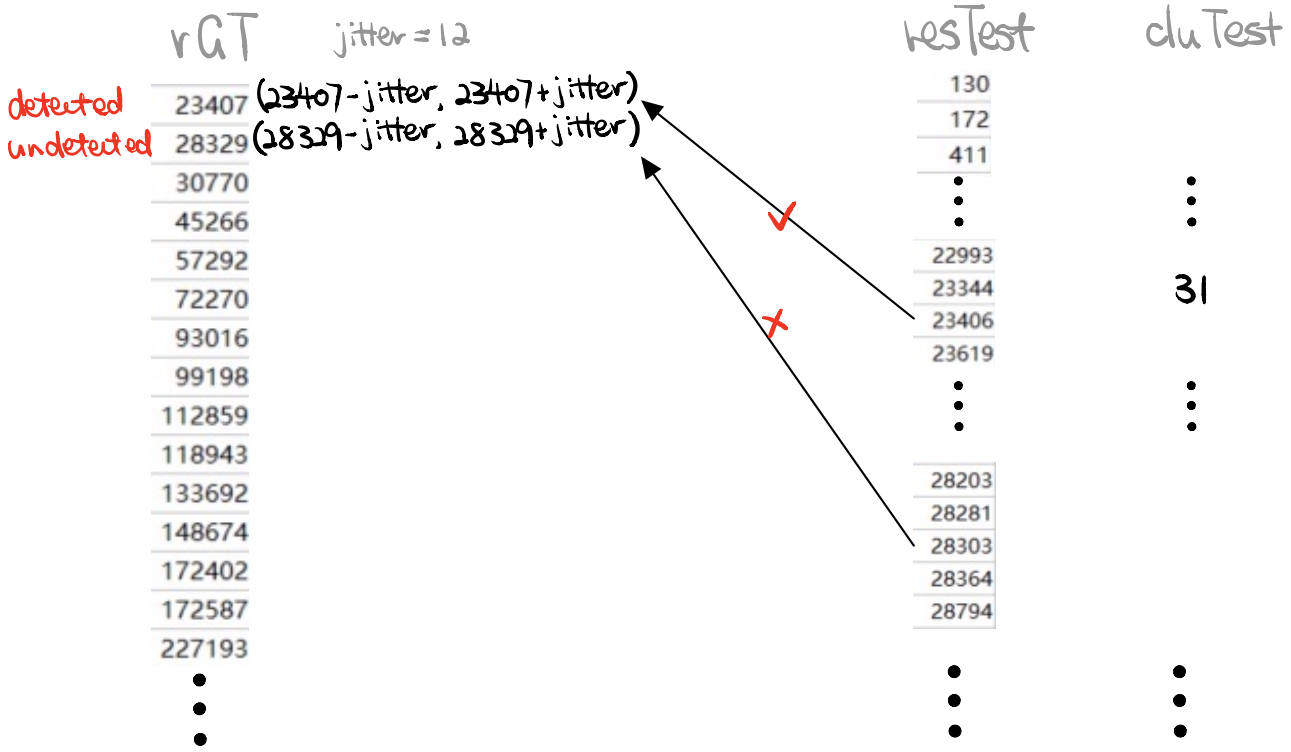
```
for cGT = 1 : length(cGTcluIDs)
```

① 得出该 neuron 的真实 spike peak time rGT

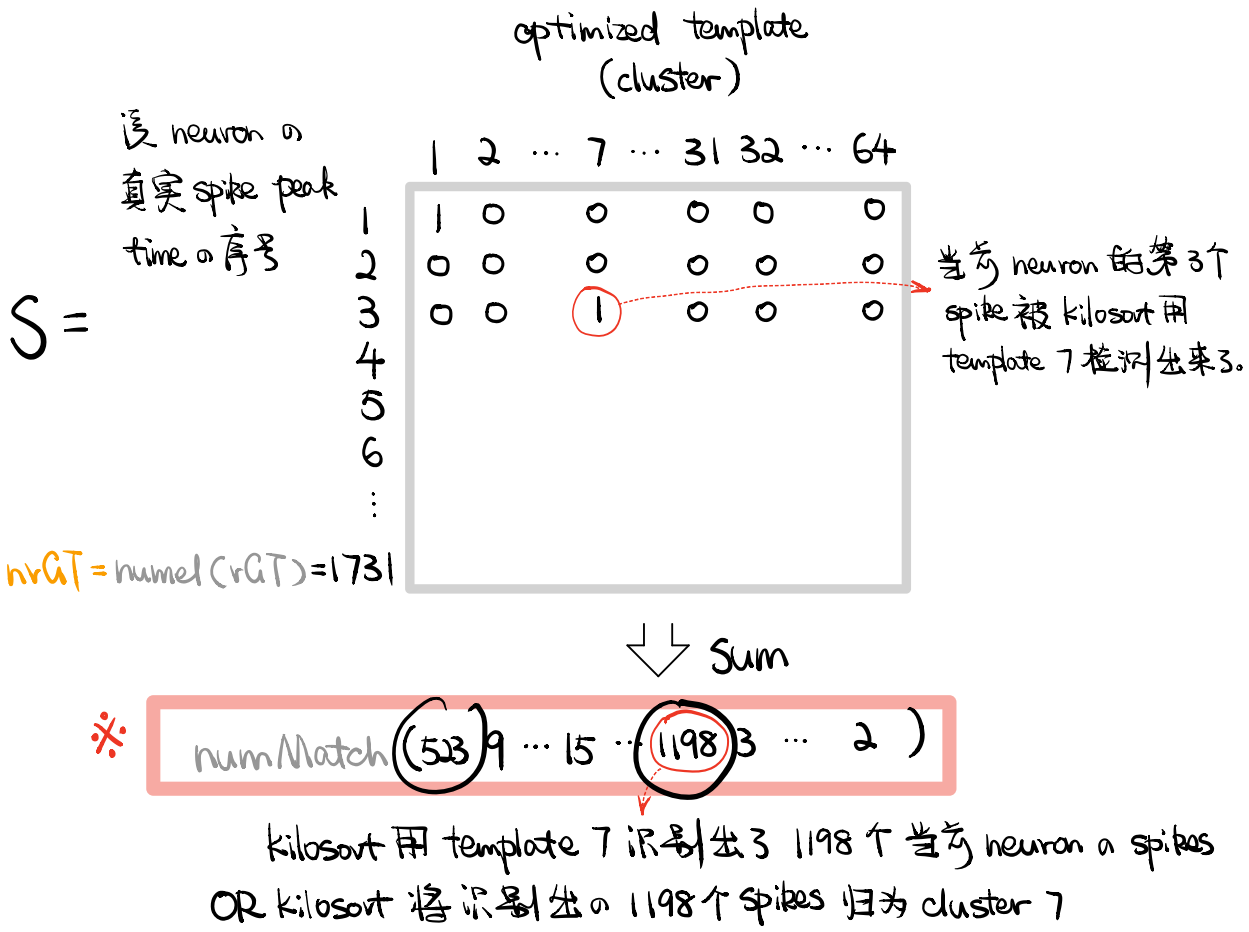
② 将所有预测的 spike peak time resTest 与各真实 spike peak time rGT 进行比较，来考察 rGT 各 spikes

是否被 kilosort 所检测。

如果被检测，kilosort 用的是哪个 template (cluster)?



将这些信息存在 S 中 (每个 neuron 都有一个 S 矩阵)

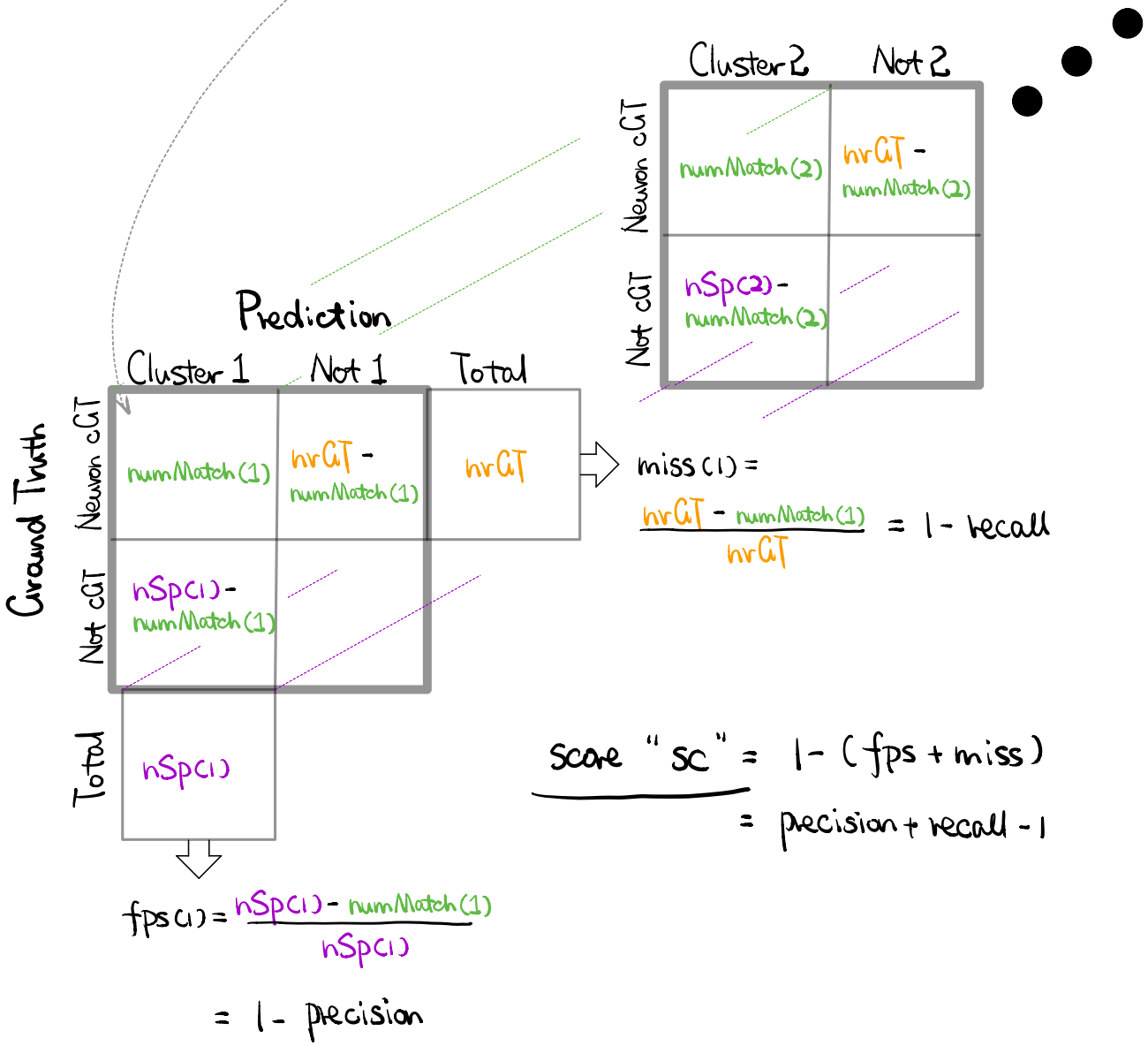


性能评估什么？我们希望将当为 neuron 的 spikes 用一个 optimized template

未匹配 (or 归类到这个 cluster 中).

假如, "template 1 (cluster 1) 是该 neuron 的 template" 成立, numMatch(1) 为 True Positive.

kilosort 将该 neuron 的 spikes 识别为 template 1 的数量



找到 sc 中的最大值, 及对应的 cluster (本例 best cluster = 7)
then 当为 neuron GT 的 spikes 就由该 cluster 的 template 来表示.
其带来的性能评分 sc(7) 最大。

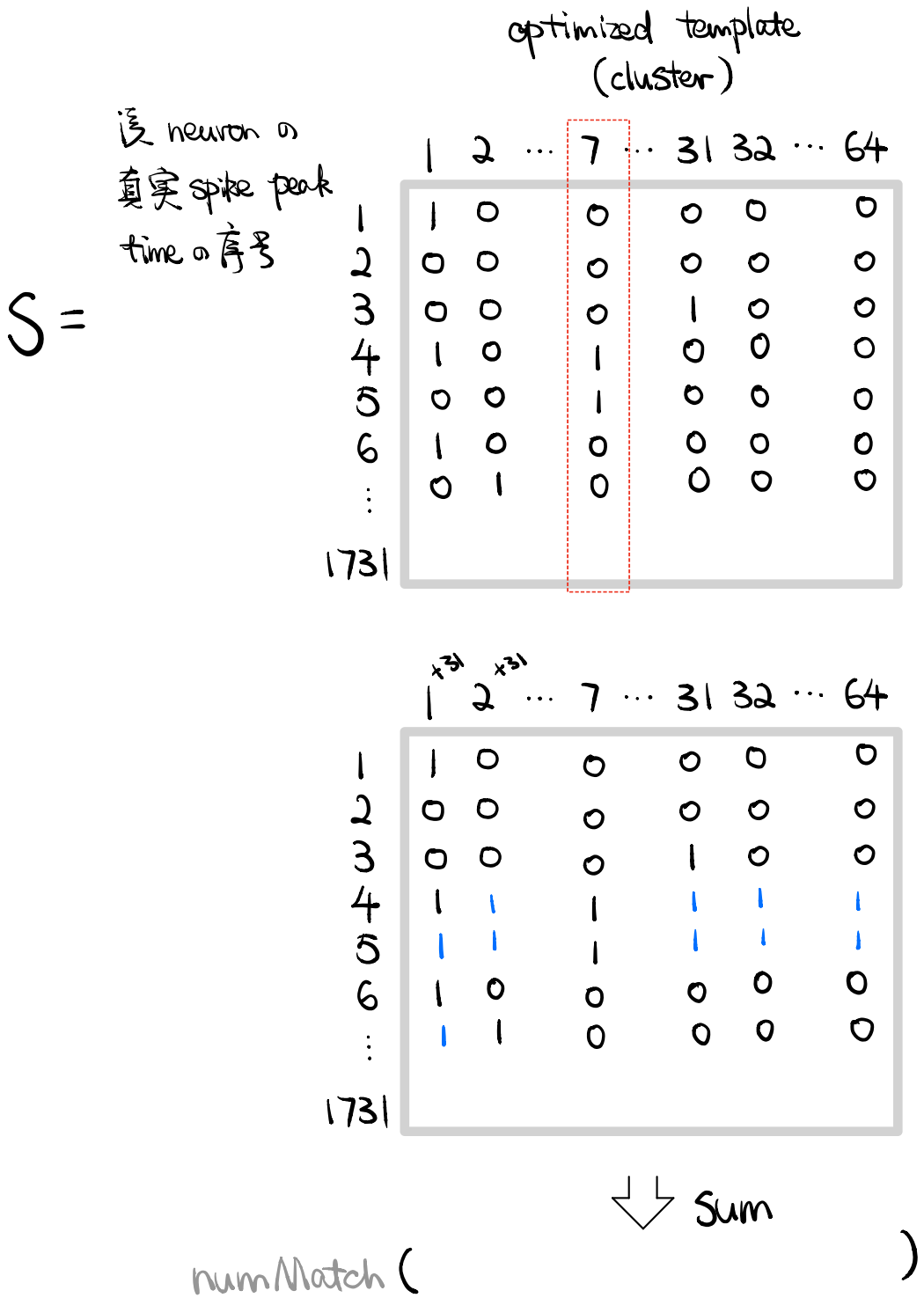
接下来, 我们尝试不止用 cluster 7, 而 cluster 7 和其它 clusters 一起来表示 neuron GT 的 spikes。我们不断将新的 cluster 与 cluster 7 merge, 来考察性能评分的提升, 从而找到一组 clusters.

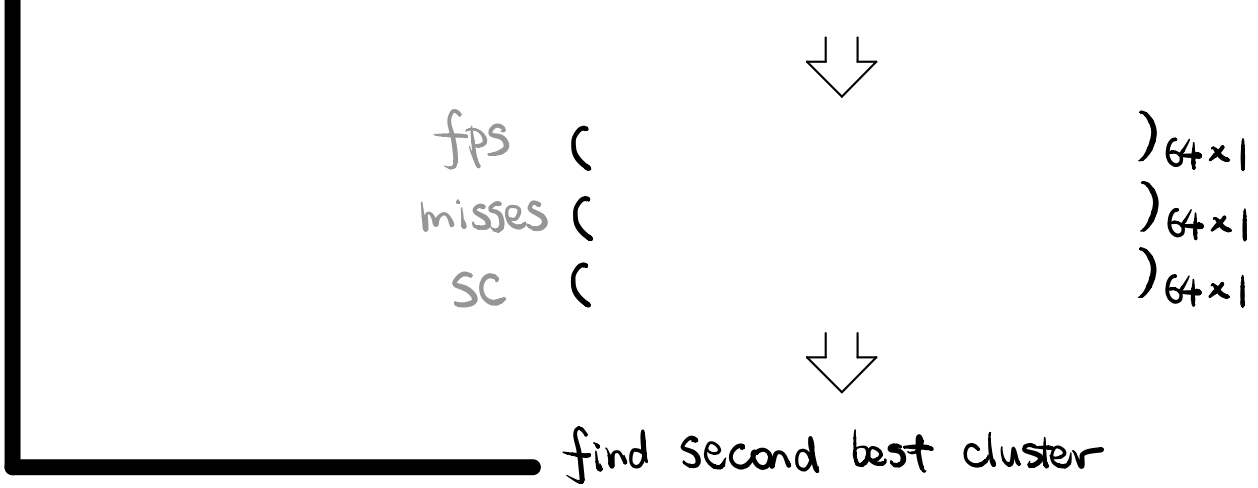
我们认为这里 cluster 具有相同 template，可以 merge.

```
while scores(end)>0 && (length(scores)==1 || ( scores(end)>(scores(end-1) + 1*0.01) && scores(end)<=0.99 ))
    % score = precision + recall - 1, since precision and recall should in the range of [0,1], score should in the range [-1,1]
    % scores(end)>(scores(end-1) + 1*0.01): after one merge, the score should has at least 0.01 increase. OR, no merge needed!
    %    for example, scores = [0.403142783662898, 0.694775145518446, 0.789568839960619, 0.850460234931639, 0.862265192017069, 0.861835060963844]
    %    the increase between end and end-1 is smaller than 0.01, so, quit the while loop,
    %    not further merge is needed for the current neuron!
    % scores(end)<=0.99: score has already reach the max, no need to merge any more
    % length(scores)==1: but if no merge has been tried, ignore the condition ( scores(end)>(scores(end-1) + 1*0.01) && scores(end)<=0.99 )

    % find the best match
    S = bsxfun(@max, S, S0);
```

相当于：第31列与每一列进行"或"运算






```
% 当前neuron算完了，存储
if length(scores)==1 || scores(end)>(scores(end-1)+0.01)
    % the last merge did help, so include it
    allMerges{cGT} = mergeIDs(1:end);
    allScores{cGT} = scores(1:end);
    allFPs{cGT} = falsePos(1:end);
    allMisses{cGT} = missRate(1:end);
else                                    output
    % the last merge actually didn't help (or didn't help enough), so exclude it
    allMerges{cGT} = mergeIDs(1:end-1);
    allScores{cGT} = scores(1:end-1);
    allFPs{cGT} = falsePos(1:end-1);
    allMisses{cGT} = missRate(1:end-1);
end
```

算完所有 neuron 之后，final step，

```
fprintf(1, '\n\n--Results Summary--\n')

for cGT = 1:length(GTcluIDs)

    initScore(cGT) = allScores{cGT}(1);
    finalScore(cGT) = allScores{cGT}(end);
    numMerges(cGT) = length(allScores{cGT})-1;
end

fprintf(1, 'median initial score: %.2f; median best score: %.2f\n', median(initScore), median(finalScore));
fprintf(1, 'total merges required: %d\n', sum(numMerges));
```

} 不存，只用于 print

allFPrates :

1x30 cell

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	[0,9.6025e-04,0.0912]	[0,0,0,9.0280e-04]	0.0236	[0.0114,0.0089]	[2.4842e-04,3.6430e-04,9.5900e-04]	[0.0110,0.0086]	5.6022e-04	0.0036	[0,0.0021]	[1.6711e-04,1.4136e-04]	0.7256	[0,0,6.4907e-04]	[0,0]	[0,1.0490e-...

allMissRates :

1x30 cell

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0.3720	[0.3284,0.1794,4.7326e-04]	[0.6682,0.3640,0.1147,9.0280e-04]	0.0529	[0.4663,0.0228]	[0.0601,0.0388,0.0269]	[0.3157,0.0115]	0.0991	0.0343	[0.0774,7.1388e-04]	[0.1555,0.0017]	0.1450	[0.4988,0.0439,8.6524e...

allScores :

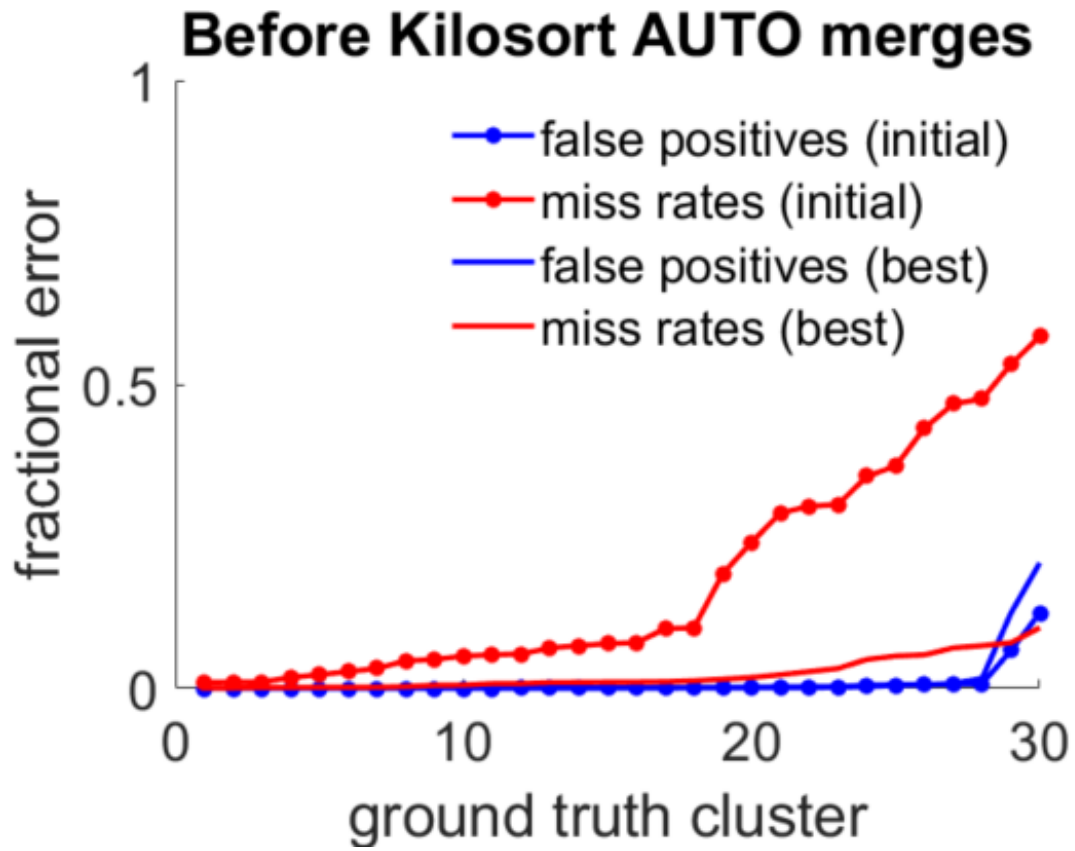
1x30 cell

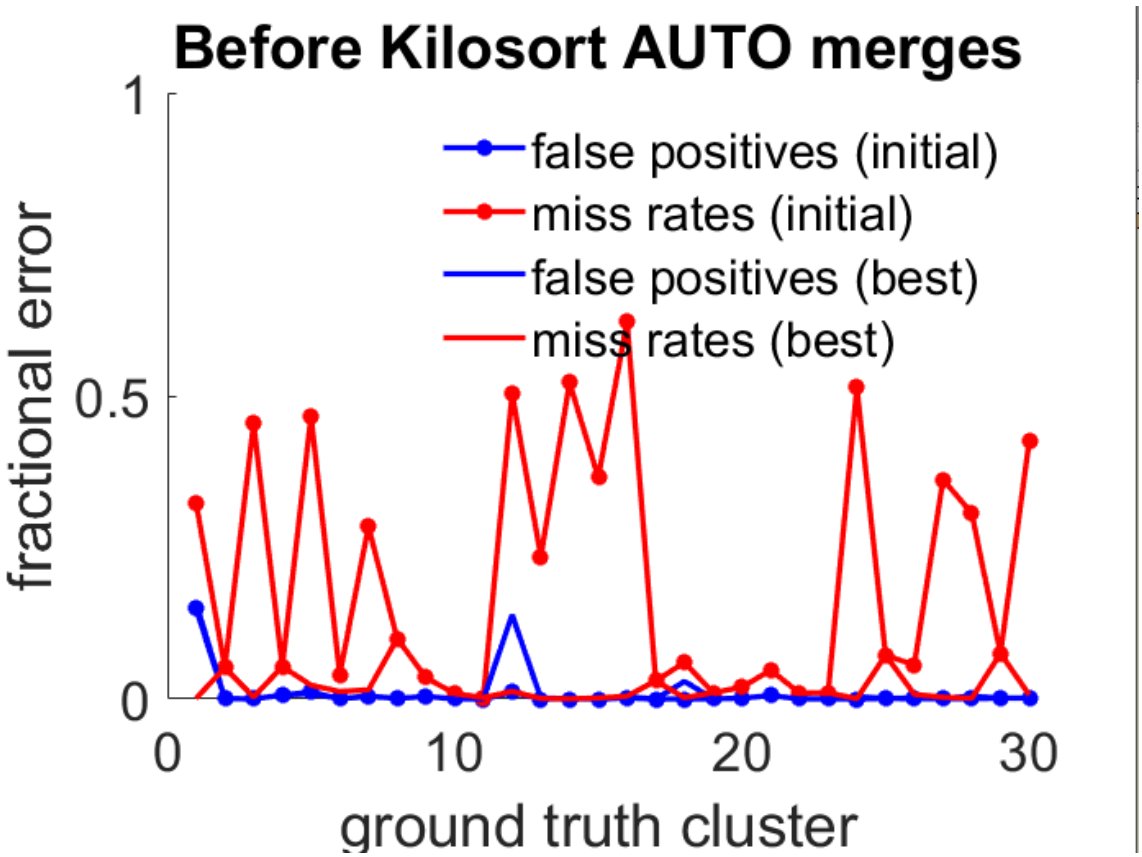
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0.6280	[0.6716,0.8197,0.9083]	[0.3318,0.6360,0.8853,0.9982]	0.9234	[0.5224,0.9683]	[0.9396,0.9609,0.9722]	[0.6732,0.9800]	0.9003	0.9621	[0.9226,0.9971]	[0.8443,0.9982]	0.1294	[0.5012,0.9561,0.9985]	[0.562

allMerges :

1x30 cell

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	53	[6,23,7]	[58,24,18,...	33	[38,8]	[35,55,28]	[44,40]	42	19	[3,48]	[22,37]	5	[36,14,39]	[13,47]	[2,56]	26	[12,50,45]	20	9	15	4




Before Kilosort AUTO merges
false positives (initial)
miss rates (initial)
false positives (best)
miss rates (best)
fractional error
1
0.5
0
0
10
20
30
ground truth cluster