# Hypothesis:

Our hypothesis is that post-processing of the random forest classifier by a random field algorithm, improves the overall segmentation, as well as the robustness.

(robustness 🡪 variance, standard.) das mit der tree methode

**Aim:**

|  |  |
| --- | --- |
| First: | * CRF algorithm (maybe combine with other filters) 🡪comparing with not post processing |
| Later: | * Try to minimize the computation time   + Number of connecting components   + Linear Programming relaxion (could be to difficult)   + Heuristic approaches (A\* Search, Uniform Cost Search Greedy search)     - How long does it take to process?     - Does it work with the complexity of the brain structure? |

* Can the performance be improved by considering the number of connected components?
* Can you think of other simple heuristic approaches that improve the performance?
* What is the **risk of post-processing segmentation**? Can you show it?
* Is the post-processing label dependent?

# Problems:

**Other stuff:**

* Mean Field Algorithm

# Experiments:

Building different Models:

* One without post processing
* CRF Algorithm
* (Model with heuristic approaches)

**Structure:**

* (Abstract)
* Introduction
  + Demonstrate importance
  + Demonstrate novelty (what has others done)
  + Present and justify hypothesis/ aim
  + Establish expectations of the report
* Methode
* Results
* Discussion & conclusion

1.b. n\_estimators :

This is the number of trees you want to build before taking the maximum voting or averages of predictions. Higher number of trees give you better performance but makes your code slower. You should choose as high value as your processor can handle because this makes your predictions stronger and more stable.

1.c. min\_sample\_leaf :

If you have built a decision tree before, you can appreciate the importance of minimum sample leaf size. Leaf is the end node of a decision tree. A smaller leaf makes the model more prone to capturing noise in train data. Generally I prefer a minimum leaf size of more than 50. However, you should try multiple leaf sizes to find the most optimum for your use case.

<https://www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/>

# Parameter explanation

## Compat:

**Kernel:**

**Normalization:**

## Pairwise Gaussian 🡪 sdims:

## Pairwise bilateral 🡪 sdims:

## Pairwise bilateral 🡪sham:

## Inferences:

# Parameter Grid Search

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Start values: |  |  |  |  |
| Compat | 1 | 5 | 10 | 25 | 50 |
| Kernel | - | - | - | - | - |
| Normalization | - | - | - | - | - |
| Paiewise Guassian 🡪 sdims | 0.1 | 0.5 | 1 | 5 | 10 |
| Pairwise bilateral 🡪 sdim | 0.1 | 0.5 | 1 | 5 | 10 |
| Pairwise bilateral 🡪sham | 0.1 | 0.5 | 1 | 5 | 10 |
| Inferences | 1 | 5 | 10 | 25 | 50 |

Evaluation: How can we get a better robustness?

These:

* Lower standard deviation 🡪 higher robustness
* Our algorithm has a high robustness if we always get better results 🡪 if we change training and test set. 🡪 other batches

Additionally:

* Some values are decreasing 🡪 optical inspection

## Questions: