Automatic Detection and Segmentation of Brain Tumor Using Random Forest Approach

Draft by Leonard Brenk

October 9, 2020

These are my first thoughts and impressions. Please correct me if I'm wrong. I specifically highlighted the parts which I am not sure about and might have misunderstood.

The idea is to enhance and improve the detection and segmentation of brain tumors whereas deciding about its presence is the primary goal of this study. Data used for training and testing decision trees and random forests were obtained from the MICCAI BRATS database. It consists of multiple volumes - meaning three-dimensional images of brains - in different types of images resulting from magnetic resonance imaging, like T1, T2, T1 post-Gladinium and FLAIR. **These types are called the features.** The images were generated from 30 glioma patients in different stages. **In this study 12 volumes were used. With four different types of volume per patient (T1, etc) there are three patients considered in this case.** Seeing that the four volumes show the same brain differently a way of addressing a single voxel - a 3D-pixel - is needed. Therefore a feature-vector has been designed to point out one specific coordinate in each of the four feature volumes:

$$\vec{x} = \Big[T1\,[x,y,z],\ T2\,[x,y,z],\ T1C\,[x,y,z],\ FLAIR\,[x,y,z] \Big]$$

Thereby we can address one voxel and compare and process it directly. This is required for training and testing binary decision trees. Before the data can be used it has to undergo several preprocessing steps like the normalization of intensity values. **The intensity of a pixel is the brightness of it.** Normalizing a histogram means that it is changed in order to locate 50% of the data between the brightness values of 600 and 800. Also, values smaller than 200 and larger than 1200 are being replaced by their limit. Another preliminary adaption of the data is the computation of location information. Since the feature vector does not carry any information regarding this, eight more features are included in the feature vector, two for each channel. **So now the feature vector contains 12 numbers.** The first is computed within a 10-element neighborhood which contains 8 elements out of the current slice and the two closest ones from neighboring slices. The second feature is based on all neighbors of the pixel in a 3 x 3 x 3 cube.

When at some point the complete data set has been preprocessed, the actual machine learning process can begin. In order for a system to recognize a pattern, binary decision trees (BDT) are being trained and tested. The intension of using such BDTs is to be able to attribute a given data vector to a class after having trained the tree. In this study, the feature vectors serve as the input data where at each node of the tree exactly one feature is being considered and allocated to either a further child node or a leaf node. While training the tree a vector that is already classified into tumor, edema, or negative by humans, is fed to the root node. In order to decide whether to go left or right a threshold is needed. Therefore we select the features that have at least two different values and sort them in increasing order. Having done so, the intermediate value is being selected as the threshold - named α - for that feature. Now the tree can forward the vector to the next node - left if that feature value is smaller than α and right if it's bigger. Wherever a decision and thereby a separation of the subset of vectors at a certain node is made, a new child node is created. At some point, it won't be possible to divide the set of vectors any further, which leads to the creation of a leaf node. Since we are still in the process of training a tree through supervised learning we already know the label of the input vector - whether the input pixel is a tumor pixel. Now we can attribute that leaf node to a certain class. That is how a BDT is trained.

A problem that can occur while training a BDT is overfitting. It's a scenario in which the tree's classification ability is optimized for that specific set of data it was trained with. For example: In a data set where all tumor pixels are a bit darker than average, the train will not detect tumors on unknown images that are lighter. The tree has not been trained for all general types of tumors an image can contain. That's why random forests are implemented. These consist - like in the real world - of many trees that were trained by random subsets of the training data. Also, the features were chosen randomly. **In the end when testing a vector, that vector is being put in all trees of the forest, which will result in many possibly different labels. Through a majority decision, the final label for that vector is then found.**
Although a random forest can detect most of the tumor pixel in an image we still can achieve a significant improvement through post-processing. Thereby we count how many neighbors of that pixel in a 250-pixel radius are also labeled the same way.