Detecting breast cancer metastases

Marc Ducret

March 2019

1 Problem setting

This project aims at building a classifier that can detected breast cancer metastases for a given patient. A patient is represented by a set of tiles $S_i = \{t_1 \dots t_{n_i}\}$ $(1 \leq n_i \leq 1000)$. Those tiles represent some small portion of tissue. Each tile can individually contain metastases. We will consider that a patient has metastases when one of their tiles does.

1.1 Tile representations

We will consider three representations of those tiles :

- Colored images of size $224 \times 224 \times 3$
- `ImageNet` embeddings via `ResNet50` of size 2048
- `PCA` representation of those embeddings of size 64

The last two representations seem sub optimal because those images are very specific and not like anything included in `ImageNet`. Also some `ResNet` features might be totally irrelevant to our problem. Since the images are quite similar to one another there are good reasons to think that `PCA` could help removing those unnecessary dimensions to reduce overfitting and computation time.

On the other hand, we can expect to extract more information from images but they have a much higher dimension which as we will see later, leads to some problems of time and memory.

1.2 Labels and objective

Some patients are *fully annotated* as each of their tiles has a label describing whether or not this tile contains metastases. For the rest of patients, only if the set of tiles contains metastases is known.

The objective is the `ROC-AUC` on patient predictions. This metric evaluates the ranking of predictions, it is maximal when all positive samples have a greater prediction than all negative samples.

2 General approach

I choose to focus on *end-to-end* architectures that take as input a stacked array of tiles. Such a model should output a global prediction and a local prediction per tile. This way, losses for both types of labels (global and local) can be used.

Since tiles are an unordered set, the architecture should respect this symmetry. Therefore a module of those architectures is the *local model*. This local model maps one tile to a prediction. Then the *global model* maps local predictions and eventually tiles to a global prediction. Since such a model can be trained end-to-end, the global loss can also help learning the local model, which should help since only few patients are annotated.

The prediction p_i associated to the set of tiles S_i can be computed using a global model $\mathcal{G}$, a local model $\mathcal{L}$ and their parameters θ:

$$p_{i,j} = \mathcal{L}_\theta(t_j)$$

$$p_i = \mathcal{G}_\theta\left((t_j, p_{i,j})_{j \leq n_i}\right)$$

Let P the set of patients, A the set of fully annotated patients[1], L_l a loss on local predictions and L_g a loss on global predictions. We optimize θ according to both losses, scaled with a factor β:

$$\arg\min_\theta \sum_{i \in P} L_i(\theta)$$

$$L_i(\theta) = L_g(p_i, y_i) + \beta \; \mathbb{1}_A(i) \sum_{j=1}^{n_i} L_l(p_{i,j}, y_{i,j})$$

3 Losses

Probabilities will be represented by their logits: $\log \frac{p}{1-p}$. The natural loss for binary classification is *binary cross entropy*. Let $L_{ce}(p, y)$ the binary cross entropy of logit p with respect to label y.

3.1 Local loss

Tile labels are extremely unbalanced as less than 1% of annotated examples are positive. Moreover, all negative patients tiles are negative. In this case, the loss must be weighted to correctly learn.

$$L_l(p, y) = w_y L_{ce}(p, y)$$

where[2] $w_y = 2\frac{\#\{i \in A | y_i = y\}}{\#A}$

This is the only local loss that was experimented with.

3.2 Global loss

The objective is the ranking of predictions. Cross entropy optimizes the accuracy and experiments showed that accuracy was not a good predictor of `ROC-AUC`.

With i_+ a positive patient and i_- a negative patient we can define a ranking loss:

$$L_{rk}(p_{i_+}, p_{i_-}) = L_{ce}(p_{i_+} - p_{i_-}, 1)$$

[1] negative patients are considered fully annotated with negative tiles

[2] $w_y = 1$ in the balanced case

Such a loss is invariant to translation of predictions, therefore, we should keep some standard cross entropy for predictions to learn this translation. To use this loss, patient must be considered in pairs. The new training objective is:

$$\arg\min_{\theta} \sum_{i_+ \in P_+} \sum_{i_- \in P_-} L_{i_+,i_-}(\theta)$$

with P_- and P_+ the respective sets of positive and negative patients

$$L_{i_+,i_-}(\theta) = L_g(p_{i_+}, p_{i_-}) + \beta \sum_{s \in \{+,-\}} \mathbb{1}_A(i_s) \sum_{j=1}^{n_{i_s}} L_l(p_{i_s,j}, y_{i_s,j})$$

Then we can define the global loss:

$$L_g(p_{i_+}, p_{i_-}) = L_{rk}(p_{i_+}, p_{i_-}) + \gamma \sum_{s \in \{+,-\}} L_{ce}(p_s, s)$$

with $\gamma \in \mathbb{R}$

4 Models

4.1 Local models

Local models mostly depend on the type of tile representation used:

- With images, some convolutional neural network should be used. However in this approach, a batch contains at least 2×1000 tiles. Even with the smallest reasonable network, this did not fit in the GPU[3]. Therefore, images were not studied in this approach.
- With `ResNet` or `PCA` features, a linear layer or a multi layer perceptron is used. One of the reasons to use `PCA` and reduce dimension, is to use a multi layer perceptron without having too many parameters[4].

4.2 Global models

4.2.1 Average

$$\mathcal{G}_{a,b}((t_j, p_{i,j})_{j \leq n_i}) = \frac{a}{n_i} \sum_{j=1}^{n_i} p_{i,j} + b$$

with $a, b \in \mathbb{R}$ parameters

4.2.2 Top-k

Let $\sigma \in \mathfrak{S}_{n_j}$ such that $p_{i,\sigma_1} \geq p_{i,\sigma_2} \geq \ldots \geq p_{i,\sigma_{n_i}}$

$$\mathcal{G}_{a,b}((t_j, p_{i,j})_{j \leq n_i}) = \sum_{j=1}^{k} a_j p_{i,\sigma_j} + b$$

with $a \in \mathbb{R}^k$ and $b \in \mathbb{R}$ parameters

The idea behind this model is that negative tiles are not relevant to the prediction. Moreover, it can learn different weights for each of the top-k predictions based on their ranking.

[3] A GTX 1080 Ti with 11 Go of memory

[4] The first layer would have $2048 \times u$ parameters for u units if `ResNet` features are used

Method	Validation	Public Test	Private Test
Average	0.79	0.77	–
Top-5	0.84	0.78	–
Attention	**0.86**	**0.80**	**0.97**

Table 1: `ROC-AUC` of different models after hyper-parameter tunning

4.2.3 Attention

Let M_θ a model that maps tiles to a logit prediction.

$$\alpha_j = \frac{e^{p_{i,j}}}{\sum_{k=1}^{n_i} e^{p_{i,k}}}$$

$$\mathcal{G}_\theta((t_j, p_{i,j})_{j \leq n_i}) = M_\theta(\sum_{j=1}^{n_i} \alpha_j t_j)$$

This model is only adapted to `ResNet` features and `PCA` since averaging of images is not desired.

5 Validation

Robust validation is necessary to select models and hyper-parameters. However, with few samples and a global metric[5] it is not an easy task.

Evaluation must be done on predictions concerning samples that were not seen during training. However, both a small training set and a small validation set lead to high variability in results. Rough validation can be obtain by training on 80% of the data and evaluating on the 20% remaining (55 samples). Different splits lead to very different scores.

A more stable approach is to make predictions on the whole training set using k-fold. Each fold splits the dataset: most is used to train the model and predictions are computed on the remaining patients. After training k models we have a prediction for each sample. Then `ROC-AUC` can be computed on those predictions. This second approach is more robust but is slow to compute.

6 Results

Table 1 shows scores of different models using `ResNet` features. The Average model uses a multi layer peceptron as local model while Top-5 and Attention use a linear model. Those local models are those that performed the best in those settings, hyper parameters such as learning rate, training epochs and weight decay were also tunned.

While there is significant variation between validation, public and private scores, the ordering seems to be the same. However, the private result of the Attention approach is very high compared to validation and public scores. While I wanted to experiment with other approaches such as CNN trained only on fully annotated patients and `PCA` with more than 64 dimensions, there would be no way to ensure that the private score is increased.

[5] Accuracy can be computed on each sample independently and then averaged but `ROC-AUC` cannot

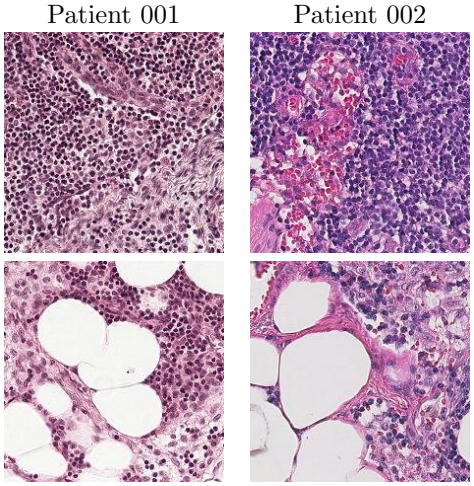

Figure 1: Color variations between patients

7 Further directions

Given a new way to evaluate models, here are some ideas to improve the model.

7.1 Tile normalisation

From a tile set to another, there are significant changes of color as seen in Figure 1. I believe this has more to do with the process of making the images than with the patient. A natural direction is to apply some normalisation to each set of tiles to remove this ambiguity. For instance, we could compute the means and standard deviations of colors over each set of tiles and normalize images this way. However, the white color is the same across all sets and should remain this way. Hence for this normalisation, only the non blank pixels should be considered.

7.2 Training a CNN

There are not many examples of positive tiles[6]. However, it seems that those examples could be easily augmented by adding rotations, flips and some noise. Training with positive-negative pairs could reduce the issues of class balance. Since images are rather simple, the network would not require too many parameters.

[6] only 707