Hierarchical, multi-label prediction for automated cell type identification

Iwijn Voeten [1] **Klaas Goethals** [1]

Abstract

This study researches the use of different high level methods to classify cells based on gene expression data. Cell types are hierarchical in nature, thus it would seem logical to use the hierarchical structure as extra data in our machine learning models.

In this paper we build different kinds of hierarchical models and compare them to regular flat classifiers where the natural hierarchy is not taken into account during training. Our findings are that hierarchical models can be trained faster than flat models while getting the same accuracy.

However, when training hierarchical models, the hierarchy must be limited to prevent the loss of accuracy. A deep hierarchical model performs worse than a regular flat classifier.

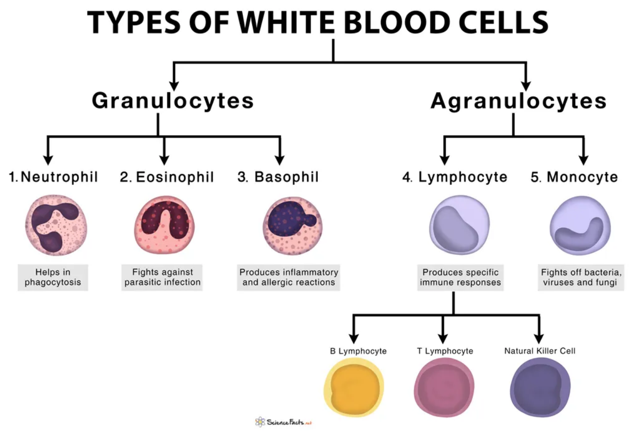
TYPES OF WHITE BLOOD CELLS
Granulocytes
Agranulocytes
1. Neutrophil
2. Eosinophil
3. Basophil
4. Lymphocyte
5. Monocyte
Helps in phagocytosis
Fights against parasitic infection
Produces inflammatory and allergic reactions
Produces specific immune responses
Fights off bacteria, viruses and fungi
B Lymphocyte
T Lymphocyte
Natural Killer Cell

Figure 1. Cell type tree of the classification made in white blood cells (Sience Facts, 2019)

1. Problem description

Cells are divided into types depending on morphological and phenotypical features. It's obvious that these types can be divided in groups, and therefore hierarchically structured. A simple example of such a hierarchical structure can be seen in Figure 1.

The problem at hand is to classify a cell based on its gene expression data. This research focuses on different hierarchical models and what their advantages and disadvantages might be.

2. Related work

A few different methods to classify hierarchical data already exist.

The most simple solution is a so-called flat, non-hierarchical classifier. This can be any multi-label machine learning classifier. It is used to classify the distinct cell types, essentially the leafs of the tree, as labels.

On the other hand, there are multiple local hierarchical models. Such a hierarchical model is a model consisting of multiple *base* models structured in a hierarchy. These hierarchic models can be divided into three main categories, as Weiss (2020) describes: **Local classifier per node**, **Local Classifier per Parent Node (LCPN)** and **Local Classifier per Level (LCL)**.

2.1. Local Classifier per Node (LCN)

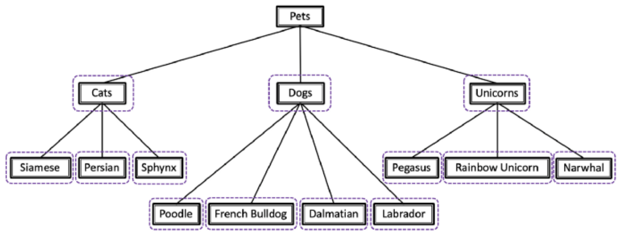

Figure 2. Visual representation of how a Local Classifier per Node model works. The circled nodes each have a classifier.

[1]Department of Applied mathematics, computer science and statistics, Ghent University, East-Flanders, Belgium. Correspondence to: Voeten Iwijn <iwijn.voeten@ugent.be>, Goethals Klaas <klaas.goethals@ugent.be>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

In this classifier, every node in the tree (except for the root) has a binary classifier. Each classifier will decide whether that specific label belongs to the given data or not.

In Figure 1 there would be 10 binary classifiers: one for every tree node except the root. A single data sample could then be classified by those classifiers as followed: the *Granulocytes*-classifier and the *Eosinophil*-classifier return a positive ('true'), all other classifiers return a negative. It can then be concluded that that sample belongs to the *Eosinophil* cell type.

2.2. Local Classifier per Parent Node (LCPN)

In this classifier, every **parent node** in the tree has a multi-label classifier. Each classifier will decide which of the node's children belongs to the given data.

In Figure 1 there would be 4 classifiers: one for every node that is not a leaf. A single data sample could then be classified by those classifiers as followed: the root classifier decides the best label is *Agranulocytes* and the *Agranulocytes*-classifier returns *Monocyle*. In this case, we *could* disregard the other classifiers and decide on *Monocyle*, but that is not necessary.

2.3. Local Classifier per Level (LCL)

This classifier has a multi-label classifier for each level in the tree. The classifier for level L can classify in each class of level L, no matter what the output is of the previous level classifier. This can however give nonsensical results.

For example, let's look at Figure 1 again. The L1 classifier could give *Granulocytes* as their output. In the next step, the L2 classifier could give *Monocyte*, a child of *Agranulocytes*, as it's output. These 2 predictions together make no sense.

In addition to these local methods, global ones exists, like Hierarchical Multi-Label Classification Networks (HMCN). This technique involves solving the problem with a neural network (Wehrmann et al., 2018). Another method is using a genetic algorithm (Cerri et al., 2012). We will leave these to the side and compare the local methods.

3. Material and methods

Multiple hierarchical methods were experimented with and tested. An attempt to parallelise a hierarchical model was also made.

3.1. Data and filtering

We had two datasets at our disposal. One is an Allen Mouse Brain (AMB) dataset. The other one is based on a BAL (Bronchoalveolar Lavage) sample of a COVID patient.

These datasets are *count tables*. They consist of *rows* that each represent a cell. For each cell, there are counts that describe which genes are expressed in the cell. So essentially the data is a table with a row representing a cell and a column representing a gene expression.

Some filtering processes were applied to this data. Columns consisting of all zeros were removed because they have no impact on the machine learning model. For example, for the "covid-BAL" dataset, approximately 14% of all columns were all zeros. In addition, rows with few non-zero numbers were removed.

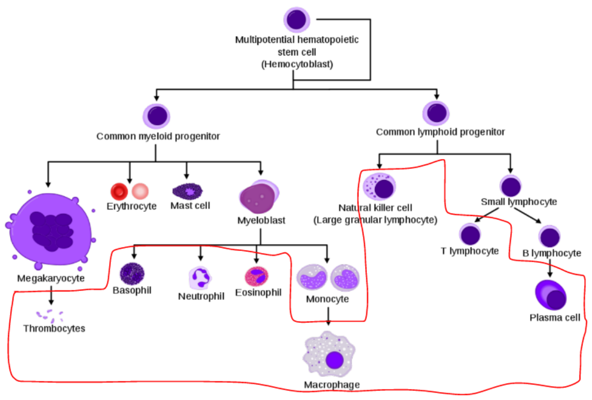
Multipotential hematopoietic stem cell (Hemocytoblast)
Common myeloid progenitor
Common lymphoid progenitor
Erythrocyte
Mast cell
Myeloblast
Natural killer cell (Large granular lymphocyte)
Small lymphocyte
T lymphocyte
B lymphocyte
Megakaryocyte
Basophil
Neutrophil
Eosinophil
Monocyte
Plasma cell
Thrombocytes
Macrophage

Figure 3. Visual representation of how a flat classifier works. The leaf nodes are the classes, the hierarchy makes no difference.

3.2. Flat classifier

A flat classifier was the baseline we compared all hierarchical models against. This is because there is no point in creating a more complex hierarchical model if the same can be done with a much simpler non-hierarchical model.

A **Linear Support Vector Classifier (SVC)**, provided by `scikit-learn` is used to classify the data (Pedregosa et al., 2011). The same kernel is also used in all below methods for consistency and because comparing different kernels is not the focal point of this research.

3.3. Local Classifier per Parent Node (LCPN)

Local Classifier per Parent Node is explained in section 2.2. A full hierarchical implementation was implemented as well as a partial hierarchical model where the depth of the tree was intentionally limited.

3.3.1. Full depth

The full depth tree works as seen in Figure 4. Each node which is not a leaf node in the tree has a multi-label classifier that can further classify the results. The con of this approach

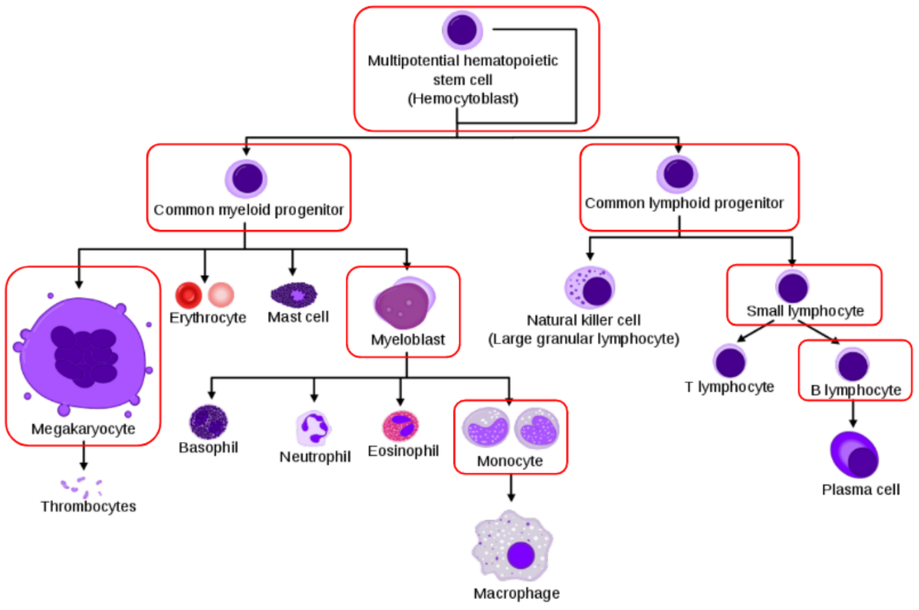
Multipotential hematopoietic stem cell (Hemocytoblast)
Common myeloid progenitor
Common lymphoid progenitor
Megakaryocyte
Erythrocyte
Mast cell
Myeloblast
Natural killer cell (Large granular lymphocyte)
Small lymphocyte
T lymphocyte
B lymphocyte
Plasma cell
Basophil
Neutrophil
Eosinophil
Monocyte
Macrophage
Thrombocytes

Figure 4. Visual representation of how a full depth hierarchical classifier works. Each circles node has a classifier.

is the fact that, for example, the *Small lymphocyte* node might have a classifier that was trained on very little data.

If only 1 percent of your dataset are cells that belong to the *Small lymphocyte* class, you can only use 1 percent of your data to train the classifier in the *Small lymphocyte* node.

3.3.2. Limited depth

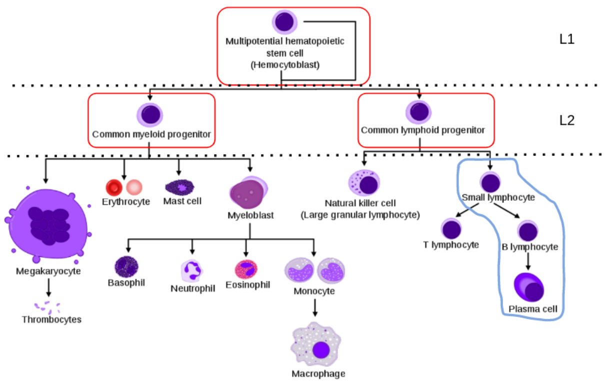

Figure 5. Visual representation of how a limited depth hierarchical classifier works. The circled nodes have a classifier. The path circled blue shows how that path is seen as 1 result that the L2 classifier can return.

The limited depth tree works as seen in Figure 5. In this model, we don't train a classifier for each non-leaf node, but only for the non-leaf that are high enough in the tree. This way, we have a bigger chance of training the classifier on a large chunk of data.

We performed tests on the Allen Mouse Brain dataset with a limited depth of 3, meaning a prediction would go through exactly 3 different multi-label classifiers to end up at its final prediction.

For example, the classifier in the *Common lymphoid progenitor* now classifies an entry into either *Natural killer cell*, *T lymphocyte* or *Plasma cell*. While in the full depth tree, it would classify into *Natural killer cell* and *Small lymphocyte*. When *Small lymphocyte* would be predicted, it would further classify with an under trained classifier.

3.4. Local Classifier per Level modified

The problems this modified classifier was attempting to solve were:

- LCL can generate nonsensical predictions
- LCPN can miss classify early on in the tree, and this is never corrected

The idea is that we sometimes want to have a nonsensical classification. When a previous classification was wrong, we want it to make a "nonsensical" classification, which chooses a different path in the tree than the wrong predic-

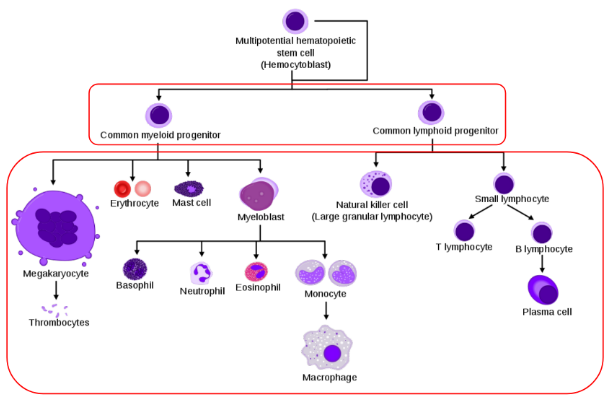
Multipotential hematopoietic stem cell (Hemocytoblast)
Common myeloid progenitor
Common lymphoid progenitor
Megakaryocyte
Thrombocytes
Erythrocyte
Mast cell
Myeloblast
Basophil
Neutrophil
Eosinophil
Monocyte
Macrophage
Natural killer cell (Large granular lymphocyte)
Small lymphocyte
T lymphocyte
B lymphocyte
Plasma cell

Figure 6. Visual representation of how a Local Classifier per Level model. Each level has a classifier. The bottom level is 1 big classifier that results in one of the leaf nodes because the depth of the tree varies.

tion.

For example, let's say we are trying to classify a *Plasma cell*. When using a hierarchical model as shown in Figure 4 a miss classification into the *Common myeloid progenitor* node would mean a definite miss classification. But when using a model like show in Figure 6, an initial miss classification into the *Common myeloid progenitor* node wouldn't make a difference for the final classification.

It might be a good idea to try and combine these two properties and create a hybrid where a previous classification can be useful information for the next classification.

This was attempted by making a classifier as shown in Figure 7. Each layer feeds its result to all following layers. This makes it so that a layer can decide for itself if it thinks one or more previous layer(s) might have miss classified.

3.4.1. Training LCL modified

Training this model had its challenges. This is because for Layer x to be trained, you need to know the output of all previous already trained layers.

To explain how a training cycle goes in this model, we will assume we are training the model in Figure 7 and thus 3 layers need to be trained. First, we divide the training set up in 3 parts. Each part will be used to train a layer. We use the first part to train the first layer as we would any multi-label classifier.

We then use this trained classifier to predict the outcomes for the second part of the dataset. These predictions, together with the gene expression data, are then used to train the second layer. As you can see in Figure 7, the second layer needs the prediction of the first layer.

Then finally we use the 3rd part of the training dataset, we feed this data to the first layer, then we feed it to the second layer together with what the first layer predicted. Now this third part has 2 extra columns, namely the prediction of the first layer, and the prediction of the second layer, this can now be used to train the third layer classifier.

This strategy should in theory now be trained to also detect when an early on classification goes wrong. The layer 2 classifier might be able to learn the right strategy when it predicts something contradictory to the layer 1 classifier.

On a tree where leaf nodes are not all at the same depth, a limited hierarchical strategy is used, similar to what is explained in section 5.

3.5. Parallelisation of hierarchical models

The hierarchical models using local methods consist of multiple models. For the LCPN technique, they should be able to be implemented with parallelisation. The local classifiers on the same level do not depend on each other. They can therefore be trained completely separate and possibly at the same time on different threads or processes.

For the LCL approaches, the same property holds: in theory the training process can be parallelised. It is however not possible for the modified LCL method we proposed. Using that strategy, every classifier depends on the ones higher in the hierarchy.

This works in theory, but in this research project we were not able to successfully implement this. The attempt to implement this with python and the `multiprocess` python library (McKerns et al., 2012) (McKerns & Aivazis, 2010) failed. The use of `Pool` to spawn multiple worker processes performs badly because the specified data is copied on each spawn. For big datasets, this makes parallelisation infeasible. To make parallelisation work, the memory should be shared in some way.

4. Results

The above methods were tested on the Allen Mouse Brain dataset using 2-fold validation. As can be seen in Table 1, no hierarchical model performed significantly better than the flat model. In fact, a deep hierarchical model performed worse than the flat classifier.

The Local Classifier per Level modified discussed in section 7 is a little bit slower than the flat classifier and unfortunately did not perform better.

A partial hierarchical model is the best option. It offers faster training, and no loss of accuracy and F1 score. On average, this method offered a 18.8% time decrease.

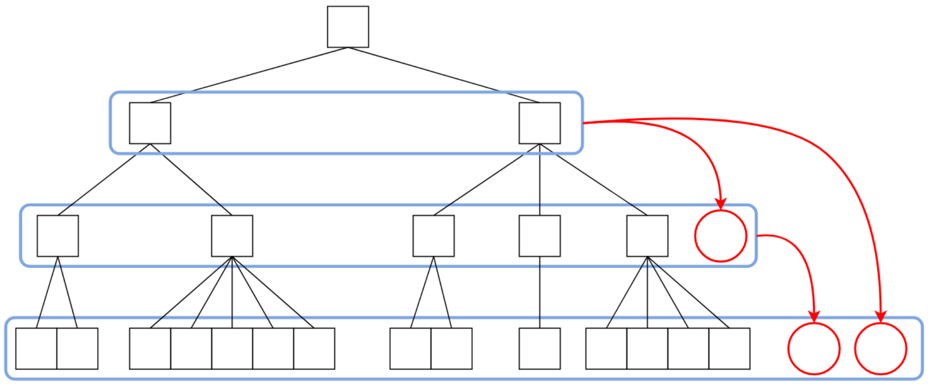

Figure 7. Visual representation of how a Local Classifier per Level modified model works. Each layer has a classifier represented as the blue circled areas. The output of each layers' classifier, represented as red circles, is given as input for the classifiers of all following layers.

Table 1. Accuracy and F1 scores weighted-average and macro-average for different models tested on the Allen Mouse Brain dataset using 2-fold validation. The limited hierarchical model had a depth of 3.

	Accuracy	F1 macro-averaged	F1 weighted-averaged
Flat classifier	0.9017	0.8421	0.8997
Limited hierarchical classifier (LCPN)	0.9020	0.8443	0.9001
Full hierarchical classifier (LCPN)	0.4720	0.6030	0.4797
LCL modified classifier	0.8673	0.7703	0.8618

In conclusion, hierarchical models can be used to speed up training, but don't offer any advantage in making a more accurate prediction.

5. Discussion and future work

Implementing the Local Classifier per Node (LCN) as explained in section 2.1 is a definite further research opportunity.

Another research opportunity is applying the limited hierarchical classifier with different depths and on many datasets. This way, a mathematical relationship might be found between the size of the data and the optimal depth of this model.

Acknowledgements

We would like to thank Lauren Theunissen for providing the two datasets. We were very grateful that the data was ready to use.

In addition, we would like to thank Daniel Peralta, Arne Gevaert, David Novak and Joris Roels for the efforts put into the practical sessions.

And finally we would like to thank Yvan Saeys for the support, feedback and theoretical lectures.

References

Abdelaal, T., Michielsen, L., Cats, D., Hoogduin, D., Mei, H., Reinders, M. J., and Mahfouz, A. A comparison of automatic cell identification methods for single-cell rna sequencing data. *Genome biology*, 20(1):1–19, 2019.

Baker, S. and Korhonen, A.-L. Initializing neural networks for hierarchical multi-label text classification. Association for Computational Linguistics, 2017.

Barros, R. C., Cerri, R., Freitas, A. A., and de Carvalho, A. C. Probabilistic clustering for hierarchical multi-label classification of protein functions. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 385–400. Springer, 2013.

Bi, W. and Kwok, J. T. Multilabel classification on tree-and dag-structured hierarchies. In *ICML*, 2011.

Cerri, R., Barros, R. C., and de Carvalho, A. C. A genetic algorithm for hierarchical multi-label classification.

In *Proceedings of the 27th annual ACM symposium on applied computing*, pp. 250–255, 2012.

McKerns, M. and Aivazis, M. pathos: a framework for heterogeneous computing. *URL https://uqfoundation.github.io/project/pathos*, 2010.

McKerns, M. M., Strand, L., Sullivan, T., Fang, A., and Aivazis, M. A. Building a framework for predictive science. *arXiv preprint arXiv:1202.1056*, 2012.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Sience Facts. https://www.sciencefacts.net/wp-content/uploads/2019/11/types-of-white-blood-cells.jpg, Nov 2019.

Wehrmann, J., Cerri, R., and Barros, R. Hierarchical multi-label classification networks. In *International Conference on Machine Learning*, pp. 5075–5084. PMLR, 2018.

Weiss, N. Hierarchical classification with local classifiers: Down the rabbit hole, Jan 2020. URL `https://towardsdatascience.com/hierarchical-classification-with-local-classifiers-down-the-rabbit-hole-21cdf3bd2382`.