Multiclass Neural Network Minimization via Tropical Newton Polytope Approximation

Georgios Smyrnis & Petros Maragos

School of ECE, National Technical University of Athens, Athens, Greece

Robot Perception and Interaction Unit, Athena Research Center, Maroussi, Greece

Spotlight

- Main problem: Minimization of a neural network.
- Various methods exist, a couple of examples:
  - (Luo et al. 2017): Removing entire neurons.
  - (Han et al. 2015): Removing connections between units.
- These methods: Remove elements from the network – more insight might be gained via the theoretical structure of the network.

Spotlight

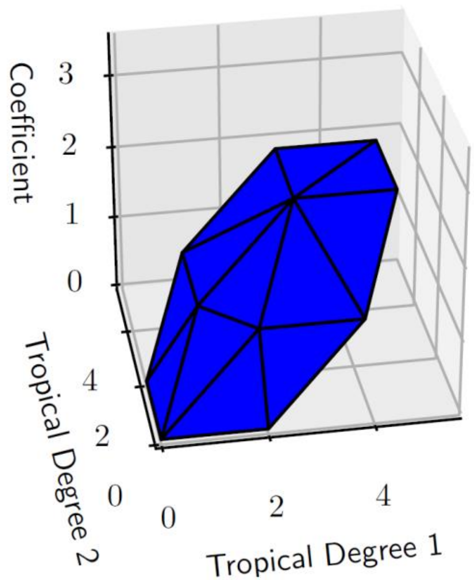
- In (Smyrnis et al. 2020): Use of tropical algebra in the domain of neural networks.
- Each network with ReLU activations: Represented by tropical polynomials (maximum of linear functions).
- Each tropical polynomial: Associated Newton Polytopes (upper hull defines polynomial).
- Tropical inspiration: Inherently linked with underlying workings of neural networks.

Spotlight

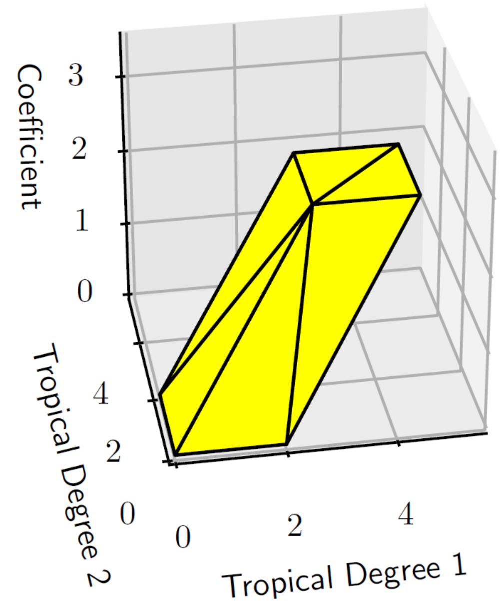
- Previously:
  - Defined approximate division of tropical polynomials.
  - Presented method for network minimization.
- In this work:
  - Extend these methods in the case of multiple output neurons.
  - Provide a more stable alternative for the single output case.

Spotlight

General idea for the task:



Original Network Polytope



Approximate Network Polytope

Spotlight

Key elements in this talk:

1. A method for a vertex transformation, to approximate the various polytopes of the network simultaneously.
2. A One-Vs-All approach, to handle each class separately.
3. A more stable minimization method for the single class case.
4. Evaluations on the minimization of pretrained networks, retaining a significant amount of the contained information.

Tropical Algebra Basics

Basics of Tropical Algebra

- Tropical algebra: Study of the max-plus semiring:

$$(\mathbb{R} \cup \{-\infty\}, \max, +)$$

- Tropical polynomial: The maximum of several linear functions:

$$p(\boldsymbol{x}) = \max_{i=1}^{k}(\boldsymbol{a}_i^T \boldsymbol{x} + b_i)$$

"Tropicalization" of a regular polynomial ($c_i \boldsymbol{x}^{\boldsymbol{a}_i} \to \boldsymbol{a}_i^T \boldsymbol{x} + b_i$).

Newton Polytopes

Let $p(\boldsymbol{x}) = \max_{i=1}^{k}(\boldsymbol{a}_i^T\boldsymbol{x} + b_i)$.

<u>Extended Newton Polytope - $\mathrm{ENewt}(p)$:</u>

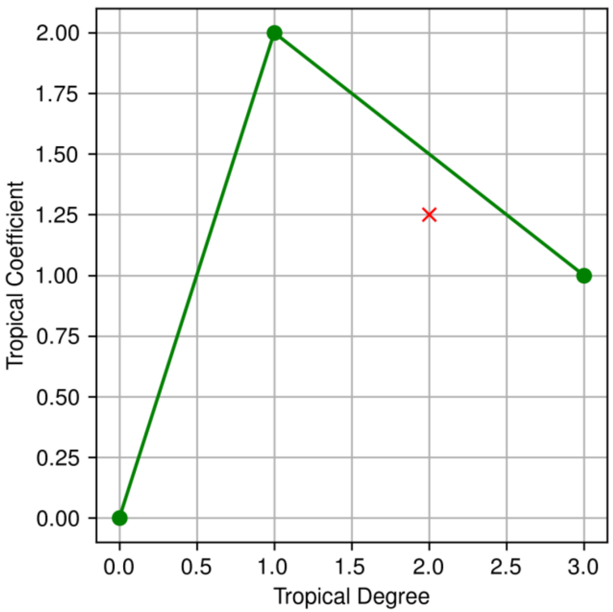
$$\mathrm{ENewt}(p) = \mathrm{conv}\{(\boldsymbol{a}_i, b_i),\ i = 1, \ldots, k\}$$

the convex hull of the exponents & coefficients of its terms, viewed as vectors.

Newton Polytopes

- "Upper" vertices of $\mathrm{ENewt}(p)$ define p as a function.
- Geometrically:
  $$\max(3x+1, 2x+1.25, x+2, 0)$$
  $$= \max(3x+1, x+2, 0)$$

(extra point is not on the upper hull).



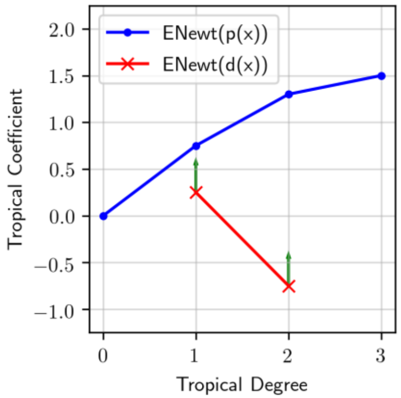
$\mathrm{ENewt}(p)$, $p(x) = \max(3x+1, x+2{,}0)$.

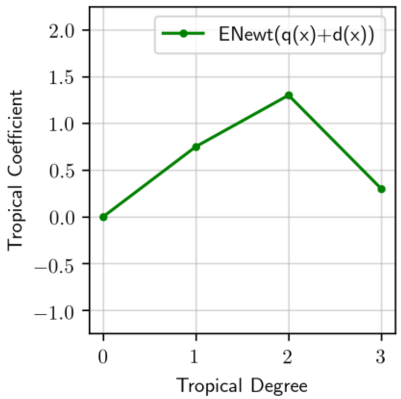
Tropical Polynomial Division

- (Smyrnis et al. 2020): We studied a form of approximate tropical polynomial division.
- We find a quotient and a remainder such that:

$$p(\boldsymbol{x}) \geq \max\left(q(\boldsymbol{x}) + d(\boldsymbol{x}), r(\boldsymbol{x})\right)$$

- How: By shifting and raising $\mathrm{ENewt}(d)$, so that it matches $\mathrm{ENewt}(p)$ as closely as possible.





Tropical Polynomials and Neural Networks

Application in Neural Networks

- In (Charisopoulos & Maragos 2017, 2018) and (Zhang et al. 2018), the link between tropical polynomials and neural networks was shown.
- The output of a neural network with ReLU activations is equal to a *tropical rational function* $p_1(\boldsymbol{x}) - p_2(\boldsymbol{x})$, the difference of two tropical polynomials.
  - Each network also has corresponding Newton polytopes.
- In (Smyrnis et al. 2020) we showed how to minimize the hidden layer of a two layer network with one output neuron, via ideas from tropical polynomial division.

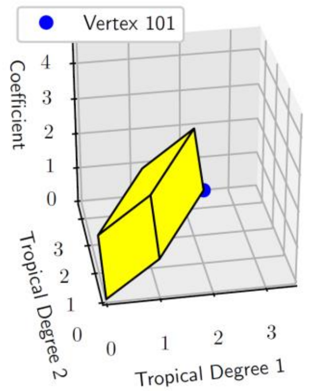
Application in Neural Networks

- <u>Main idea of (Smyrnis et al. 2020):</u>
  - Find a divisor which approximates the polytopes of p_1, p_2:
    1. Calculate the "importance" of each vertex.
    2. Add the first vertex as a neuron.
    3. Add as a neuron the difference of each new vertex from a random previous one.

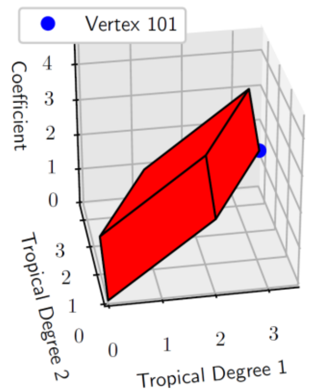
    Intuition: Sums of neurons become polytope vertices.
  - Set the average difference in activations as output bias (quotient).
- In the following, we shall refer to this as the heuristic method.

Application in Multiclass Networks

Extension with Multiple Output Neurons



Upper hull of polytope, Neuron 1



Upper hull of polytope, Neuron 2

- <u>What we have:</u> Multiple polytopes, interconnected (as seen in the figure).
- <u>What we want:</u> Simultaneous approximation of all polytopes.

Binary Description of Vertices

- The polytopes of the network are *zonotopes*: they are constructed via line segments (each corresponding to one neuron).
- Each vertex has a natural binary representation: the neurons corresponding to the line segments it is constructed from.
- Vertex weight: The sum of the respective neuron weights.
- Previous figure: The polytopes of the output neurons share the binary representation.

First Method: Approximation with a Vertex Transform

For an output neuron with weights $\boldsymbol{w}_l^2$, and a hidden layer with weights $\boldsymbol{W}^1$, a vertex of the polytope can be represented as:

$$\boldsymbol{v} = \boldsymbol{W}^1 \mathrm{diag}\left(\boldsymbol{w}_l^2\right) \mathbf{1}_{\boldsymbol{v}}$$

where $\mathbf{1}_{\boldsymbol{v}}$ is a binary column vector of the representation.

First Method: Approximation with a Vertex Transform

- The method is as follows:
  - Perform the single output neuron minimization, assuming all output weights are equal to 1.
  - For each output neuron, find the original representation of the chosen points $\boldsymbol{v}$.
  - Using the new weight matrix $(\boldsymbol{W}^1)'$, find the optimal weights for the output layer, so that:

$$\boldsymbol{v}' \approx \boldsymbol{v}$$

  - Add the output bias as before.
- However, this treats all classes in the same fashion: <u>counter-intuitive!</u>

Second Method: One-Vs-All

- Second approach: treat each output neuron (class) separately:
  - Copy the hidden layer once for each output neuron.
  - Minimize each copy with the single output neuron method.
  - Combine all reduced copies in a new network.
- To rank the importance of a sample: reweighting.
  - C output classes: positive samples count as $C - 1$.
  - Negative samples for each class count as 1.

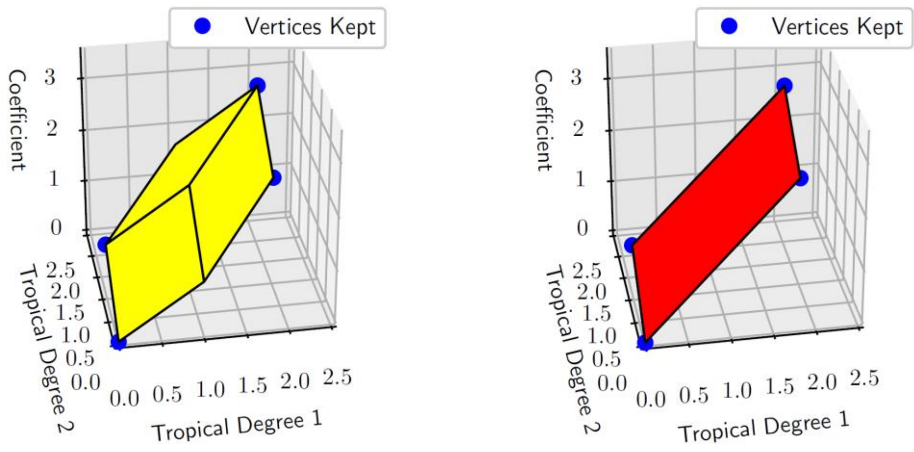
Alternative Method for Single Output Neuron

Alternative Method for Single Output Neuron

<u>Outline of the algorithm for the divisor:</u>

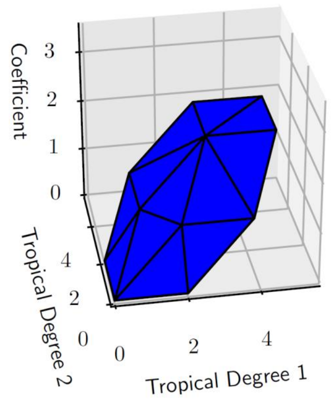
- Calculate the importance of each vertex as before.
- Convert each vertex to its binary representation.
- Add new vertices, splitting their binary representations so that each neuron of the original hidden layer is contained at most once.
  - Example: Vertices 1110, 0111 - three neurons: 1000, 0110, 0001.
  - This way, new vertices are strictly inside the original polytope.
- Find the actual weights of the final neurons (via binary representation).

Alternative Method for Single Output Neuron

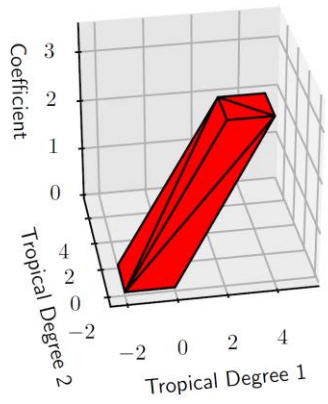


- Final polytope (right) is precisely under the original (left).
- The process is a “smoothing” of the original polytope.
- It is deterministic: less variation is expected.
- Extra output bias: Average difference in activations (to address samples not covered by chosen vertices).

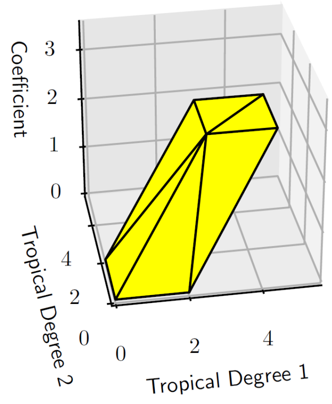
Properties of the Stable Method



Original Polytope



Approximate Polytope, Heuristic Method



Approximate Polytope, Stable Method

1. Approximate polytope of the divisor contains only vertices of the original.
2. The samples corresponding to the chosen vertices have the same output in the two networks (without the extra output bias).

Properties of the Stable Method

3. At least:

$$\frac{N}{\sum_{j=0}^{d} \binom{n}{j}} O(\log n')$$

samples retain their output (N is the number of samples, n and n' the number of neurons in the hidden layer before and after the approximation). Note that this is not a tight bound.

Experimental Evaluation

Experimental Evaluation

- We evaluate our methods on two datasets:
  - MNIST Dataset
  - Fashion – MNIST Dataset
- The architecture in both datasets consists of:
  - 2 convolutional layers, with max-pooling.
  - 2 fully connected layers.
- For each trial, we minimize the second-to-last fully connected layer, with the One-Vs-All method.
- Results: Average accuracy and standard deviation for 5 trials.

MNIST Dataset

Neurons Kept	Heuristic Method, Avg. Accuracy	Heuristic Method, St. Deviation	Stable Method, Avg. Accuracy	Stable Method, St. Deviation
Original	98.604	0.027	-	-
75%	95.048	1.552	**96.560**	**1.245**
50%	95.522	3.003	**96.392**	**1.177**
25%	91.040	5.882	**95.154**	**2.356**
10%	92.790	3.530	**93.748**	**2.572**
5%	92.928	2.589	92.928	2.589

Fashion-MNIST Dataset

Neurons Kept	**Stable Method, Avg. Accuracy**	**Stable Method, St. Deviation**
Original	88.658	0.538
90%	**83.634**	2.894
75%	**83.556**	2.885
50%	**83.300**	2.799
25%	**82.224**	2.845
10%	**80.430**	3.267

Conclusions & Future Work

Conclusions & Future Work

- In this work:
  - We extended work done in (Smyrnis et al. 2020) to include networks trained for classification tasks with multiple classes.
  - We presented a stable alternative to the method in (Smyrnis et al 2020).
- Moving on, we will try to:
  - Extend these methods in more complicated architectures.
  - Evaluate them in comparison with existing minimization techniques, in more complicated datasets.

References

- S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural network”, in *Advances in Neural Information Processing Systems 28*, 2015, pp. 1135–1143.
- J.-H. Luo, J. Wu, and W. Lin, “ThiNet: A filter level pruning method for deep neural network compression”, in *Proc. Int'l Conf. on Computer Vision*, Oct. 2017.
- G. Smyrnis, P. Maragos, and G. Retsinas, “Maxpolynomial division with application to neural network simplification”, in *Proc. ICASSP ‘20*, IEEE, 2020, pp. 4192–4196.
- V. Charisopoulos and P. Maragos, “Morphological Perceptrons: Geometry and Training Algorithms”, in *Proc. Int’l Symp. Mathematical Morphology (ISMM),* ser. LNCS, vol. 10225, Springer, Cham, 2017, pp. 3–15.
- V. Charisopoulos and P. Maragos, “A tropical approach to neural networks with piecewise linear activations”, *arXiv preprint arXiv:1805.08749*, 2018.
- L. Zhang, G. Naitzat and L.-H. Lim, “Tropical geometry of deep neural networks”, in *Proc. Int’l Conf. on Machine Learning,* vol. 80, PMLR, 2018, pp. 5824–5832.

THANK YOU FOR YOUR ATTENTION!

For more information, demos, and current results, visit:

http://cvsp.cs.ntua.gr and http://robotics.ntua.gr