1 Motivation

I propose a more biologically accurate approach for the regularization of ANNs by adding a new parameter for every neuron to be considered, better representing true hebbian-learning. Like the biases, this parameter is initialized with a vector of zeros the same size as the number of neurons. For example, if using the RELU activation function, every time signals are sent to a neuron, the hebbian value is incremented by the output of the function if it is greater than 0 and decremented by a constant value if the neuron is not activated. This constant is an hyper-parameter: if the value is too high, it risks to remove too many neurons. If it is too high, neurons might not be pruned enough for an efficient generalization of the network learned.

1.1 Hebb's principle

Based on Hebb's principle, which suggest that "Neurons that fire toghter, wires togheter", Hebbian mechanisms in artificial networks are thought to be represented by the weights of the propagation functions. The weights are learned through backpropagation, with the use of gradient descent. Instead of representing Hebbian mechanisms, I suggest this form of learning might better represent Non-Hebbian mechanisms, because it doesn't depend on the contribution of the presynaptic synapse, but rather solely on the activation of the postsynaptic neurons. In true Hebbian mechanisms, for synapses to keep existing, they need to be used at a minimum frequency, or they die. Dropout is probably getting artificial neural networks (ANNs) closer to true Hebbian mechanisms, as it prevents over-learning some features that are too specific to the example and, consequently, do not generalize well. Although Dropout reduces the importance learned features (weights and biases) can take, thus forcing to learn a more generalizable model that doesn't rely too much on a few features, some features might still be over-learning some features. I propose two processes that, when added to the already existing ANNs training processes, would result in a better form of learning Hebbian that respect Hebbian mechanisms.

2 Hebbian mechanisms in human brain development

2.1 Long-Term Potentiation (LTP) and Long-Term Depression (LTD)

In biological neurons, learning is mainly driven by the long-term potentiation (LTP) and long-term depression (LTD) of synapses. LTP is a process that increases the strength of the synapses and is thought to be driven by increasing the number of voltage-dependant proteins (e.g. NMDA receptors) in the postsynaptic membrane of synapses with increasing action potentials being released; LTD is the opposite process. Hebbian-LTP require both the pre-synaptic and post-synaptic neurons to be simultaneously activated to increase the synaptic strength. Accordingly, Hebbian-LTD require the pre- and post-synaptic neurons involved in a synapse to both get reduced at the same time. In ANNs,

2.2 Hebbian vs other types of learning

In biological neurons, it was discovered that LTP and LTD can be driven by other processes, namely Non-Hebbian and Anti-Hebbian. Interestingly, non-Hebbian mechanisms is not "depend on the timing between presynaptic and postsynaptic activity, but it could be induced by postsynaptic burst activity alone" (Sieber,2013). This definition would better describe ANNs learning through backpropagation than Hebbian mechanisms. While backpropagation is very efficient in actual ANNs, they need to be regulated so they don't over-learn some examples, which makes them inefficient on test samples. In the brain, the development of neural networks is effectuated by first adding new neurons to increase the learning capacity. Those new neurons are added "en masse" during development , but useless neurons also die at a fast rate. This process adds neurons randomly (though some regions might be more targeted at different age periods) and only those that were used to learn a new feature are kept, independent of if this feature is actually true.

3 Methods for ANNs

3.1 Main idea

Neuron connections are kept in a neural network only on the basis of how much they were use, ignoring how useful it could be.

3.2 Hebb values

Each neuron possesses a Hebb value. It could be imagined as the cell's "stamina".

During training, every time a sample is fed to the network, all Hebb values are updated and used in the next sample. My implementation can only use Relu as the activation function.

- If the neuron fires (activated), the post-activation value is added to the neuron's Hebb value.
- if the neuron doesn't fire, instead of adding the post-activation value, which would be 0, a fixed number is substracted from the Hebb value. The pruning rate would depend on this fixed number. If the rate is too high, everything will be removed. It could also prevent the network from growing when it could benefit to grow. Too low and nothing will ever get removed. If also adding neurons, the network could grow forever.

Hebb values are initiated at 0. It could be different, but it has to be the same for all neurons.

3.3 Apoptosis

At the end of each epoch, I remove any neuron with Hebb values smaller than a certain threshold. The threshold can be modified to let the network learn something before deciding to remove any neurons.

3.3.1 Initialization of new neurons

New neurons are initiated using Glorot's Uniform initialization. If 10 neurons are added to a 90 neurons layers, then the 10 new values are obtained randomly as part of a 100 neurons layer.

I also implemented (but did not maintain) a way to get new neurons that would be replicates of the neurons already existing (all of them or a fraction). I think it was functional. Gene duplication is a common way for a new gene with a distinct function to be formed. I think has evolution has another form of learning, and duplication would be a valid way to initiate a new gene to be learned through a very long time, so I taught it might be worth exploring if it also applied to ANNs. Also, being a valid initialization method would not make it better.

3.3.2 Pruning neurons to improve Generalization

Hebb values have no impact on feature learning itself. It only indicates when a neuron is not considered active enough to be kept. There are some exceptions, but because an intelligent system (e.g. an animal with a central nervous system or a machine that would constantly be learning) is limited in the amount of information it can hold, it needs to somehow keep track of the importance of each neurons and which ones could be erased with a minimum loss of information.

3.4 Neurogenesis

New randomly initiated cells are added to each layer.
Neurons will only be added when the current ANN is struggling to improve and could benefit from having new neurons.

3.5 Hyper-parameters: Determination of the ANNs growing/pruning rates

In theory, because the neurons that are forced to be used a minimum of times, this should help the regulation of the network. A mature network (one that has seen many epochs) with the same architecture (# neurons, etc...) than than a more immature network should have learned a model that generalize better than the latter. The more immature one could be as good or even better than the more mature network. The latter would have learned through experience (has seen more examples) that some features in the training are misleading and not generalizable.

In addition to having to determinate a minimum hebb value and their initial constant value, new hyperparameters are needed:

-
- Values to remove in the case of no activation
- Growing rate (how many neurons to add when needed)

3.5.1 Advantages

- The number of neurons in all layers depends on the task to learn. Won't be too big for no reason. I don't know if it is possible to add layers though without breaking what has already been learn, so this part of the architecture has to be predetermined.
- It should help avoiding overlearning (on top of the other existing ones). A mature network would typically have been through multiple network growing/pruning processes. With time, the neurons that are kept would have to be less specialized. Here is how I see it:
  Let's assume we have a single-layer network with X neurons. That network cannot converge unless another neuron is added. Depending on how the weights and biases are initiated, a new neuron could later get to many possible local minima, but only some initial value could eventually lead to convergence. If done correctly, a few neurons would be added to get it right and if a value leading to convergence is found, it should be kept and the rest would die.
- It could help to reduce the total amount of time required to train a neural network.
  I am not sure it would be true at the moment if it would be true if the optimal number of neurons is already known. it depends how fast the network is allowed to grow; it might be faster because the earlier steps will be much faster, because the initial number of parameters will be much smaller then at the end. If it took the same number of epochs it would clearly be faster. I did not have the opportunity to test many combinations of hyper-parameters. In the end, if one would have to test with the number of neurons per layer, this automation would reduce the number of settings to try.
- It will learn more meaningful models earlier than having an very large initial network
  Smaller networks can sometimes be almost as good as their larger counterparts. A very big difference in the number of total parameters in the models might only result in a relatively slim difference in prediction accuracy
- As a network progress, it might adopt complex shapes that might never be tried by a user. The number of possibilities if very high and would not usually be practically feasible to try even most of them (observation of mine; needs more investigation) .

This method should let the network develop itself to learn the best network architecture it can given limitations in size.

4 Divergences of Hebb Values from the biological inspiration

Just like many other ANNs methods that have features inspired by biological neural networks, some elements in this model are deliberately diverging from the biological neural network analogy. In my case, the differences are only motivated for the sake of computing efficiency.

4.1 Comparison with biological neural networks

4.1.1 Main stages of the human brain development

- The brain is initialized with some intrinsic configuration and knowledge before birth, that is even before getting the opportunity to learn anything. This much have been learned through millions of years of evolution and is a crucial part in the success of the human brain. Interestingly, ANNs initialization is now known to be very important to successfully train deep networks. Both are similar in the sense that they both need a good start to perform optimally, but biological networks seem to be much more influenced by their initial state. They can learn, but ANNs are most probably more plastic. Birth would be the end of a biological neural network "initialization".

- Immediately after birth, the brain grows rapidly, with a lot of new neurons being generated (neurogenesis), but also deleted (apoptosis) when new connections are not useful enough. My hypothesis is that biological neurons not necessarily learn from their mistakes, otherwise. For example, it should be easier for someone to correct its own behaviours he/she consider to be wrong. Another example: jingles or slogans are not so easily unconsciously remembered for their usefulness, but rather because it was heard frequently.

- During adolescence, neurogenesis slows progressively and stops almost completely during early adulthood (with some exceptions, mainly zone subventricular zone of the striatum and the subgranular zone of the hippocampus).

- Finally, the number of neurons in the network will only go down for most parts until death. New connections can still be made during this time though, maintaining the brain's capacity to learn, but at a much lower rate.

After birth, I would summarize the brain's development as follows:

- Fast learning stage. During that time, new concepts are rapidly and easily assimilated. At this stage, the brain is mostly good at repeating what it was taught, but higher concepts take more time to understand. In this part, there would be susceptibility to over-learning.

- Understanding stage (confusing part, also known as adolescence). The brain learns to better understand concepts it already understand.

- Fine-tuning stage. In adulthood, the neurons are mostly eliminated, slow it is a slow decay until the end (sorry if this sounds depressing!).

4.1.2 Main differences with Hebb's principle in neurobiology

- LTP and LTD actually apply specifically to the synapses individually. In ANNs, I am adding values from all pre-synaptic neurons (previous hidden layer) to get a score for the post-synaptic neuron; post-synaptic neurons are dropped if not activated by pre-synaptic neurons. Thus, all connections (represented by weights) in neurons are kept or pruned in my model. The reason for keeping it this

way is to reduce computer resources; it would be very expansive computationally to keep track of all of them.

- New neurons are added when needed. In a growing brain, neurogenesis would be more chaotic (but, from what I understand, it is true for everything in ANNs).
- In a normal biological human brain development, the number of cell would grow fast and continually with new neurons getting integrated to the rest of the existing network. It would get a basic architecture from birth, with implicit knowledge in the initial network (babies know how to cry, etc...), but would then grow and develop differently. What is learned depends on the environment (teachers, wealth, physical strength, sex, dangers, influences, etc).

5 Alternative directions

5.1 Replacing the neurons with a fixed number of neurons per layer

I have designed a way to keep the number of neurons stable. Simply, neurons with Hebb values that are too are not removed, but instead reinitiated. I have implemented its, but I did not test them as much and it might not be functional anymore.

5.2 Synaptic Hebb values

A more complicated approach might also improve the method I propose, but based on the same idea. Instead of assigning an single Hebbian value for each neuron, a value could be assign to every axon individually. This would invalidate axons individually and a neuron would die only when all its axons are dead. This approach would reflect the development of biological neural networks even better.

6 Note on the biological inspiration

Backpropagation could be seen as modifying the presynaptic signal (axons) and the Hebbian mechanisms I propose as the number of post-synaptic sites available (neurites) available to connect with. We will suppose that as long as there is a single neurite available, all axons can connect to it and activate the post-synaptic neuron just as efficiently as if their is a lot of them. However, when there would no longer be any neurites left, the presynaptic cells cannot connect to it anymore and the postsynaptic cell dies.