Training neural networks: the nitty-gritty

We know that the weights $\mathbf{W}^*$ that minimize the average loss

$$\mathbf{W}^* = \underset{W}{\operatorname{argmin}}\, L(\hat{\mathbf{y}}, \mathbf{y}; \mathbf{W})$$

can be found by using Back Propagation (see the earlier module (Training_Neural_Network_Backprop.ipynb)) to compute the derivatives needed by the Gradient Descent algorithm.

So training a Neural Network sounds simple enough.

But we also recalled several "AI Winters" in which progress in Deep Learning stalled.

History indicates that Training is more complex than it appears.

We now spend some time investigating the causes, and solutions, to the difficulty of training networks.

Broadly speaking the issues are

- Gradients becoming zero or infinity, inhibiting learning (weight updates in Gradient Descent)
- Proper scaling of the inputs
- Initialization of learnable weights
- Making sure that the proper scaling of inputs continues to each layer, not just the input

Vanishing and Exploding Gradients

Although Backpropagation is mechanically simple, there are some mathematical subtleties.

Let's explore the problem of Vanishing and Exploding Gradients (Vanishing_and_Exploding_Gradients.ipynb)

Initializing and maintaining layer inputs

Neural Networks are sensitive to the scale of the layer inputs.

Creating the correct situation to learn is the subject of Scaling and Initialization (Training_Neural_Networks_Scaling_and_Initialization.ipynb).

Improving trainability

Apart from the mathematical issues of preventing activations and gradients from exploding/vanishing, there are many ways to make training successful.

Let's explore techniques for Improving trainability (Training_Neural_Networks_Tweaks.ipynb).

How big should my NN be ?

There is a paradox in building Neural Networks:

- Start off training an overly large NN (many units)
- Many units turn out to be "dead": near zero weights
- Reduce the number of units
- Can't train !

Given a fixed number of layers: it is easier to train a big NN than a small one.

"Somewhere in this big mess must be something valuable"

"Big" NN with dead nodes

The Lottery Ticket Hypothesis (https://arxiv.org/abs/1803.03635) is an inter
that addresses this issue.

For now:

- Use bigger than necessary NN's
- With regularization to "prune"

Conclusion

We sometimes take training Neural Networks for granted.

After all, Gradient Descent seems like a simple procedure.

It turns out that there are *many* subtleties.

Uncovering and solving the subtle problems were the key contributions in rapid advance of Deep Learning.

Without them, we'd still be living in "AI Winter".

In [3]:

```
Done
```

"Big" NN after dead nodes have been pruned