Improving Generalization in Neural Networks Through Margin Maximization

Siddartha Devic, Dr. Nicholas Ruozzi*
The University of Texas at Dallas

I. What Are Neural Networks?

Recent developments in computer hardware have enabled a specific part of the computer, the Graphics Processing Unit (GPU), to accelerate developments within a sub field of artificial intelligence: machine learning. Specifically, GPU's have enabled computers to quickly create accurate prediction systems known as "neural networks"—previously a computationally infeasible task.

A neural network comprises of thousands of nodes and weighted connections between these nodes that attempt to accurately predict an output given a certain input (Figure 1). Given a certain input, like an image of a cat or dog, it will predict an output, such as "cat" or "dog". A neural network's lifespan is divided into two parts: training and inference.

During training, the network passes the images of a human-labeled dataset through itself, guesses the label of each image, measures its accuracy, and attempts to minimize the error of its guesses. It does this thousands of times, which is why training is a time consuming process.

In the inference stage, we present to the network images not previously seen in the training phase. Remarkably, it is able to classify them with a very high degree of accuracy. Why, after training, does the network give the correct classification when it is shown an image of a cat or dog it has never seen before? This is the so-called "generalization mystery" of neural networks.

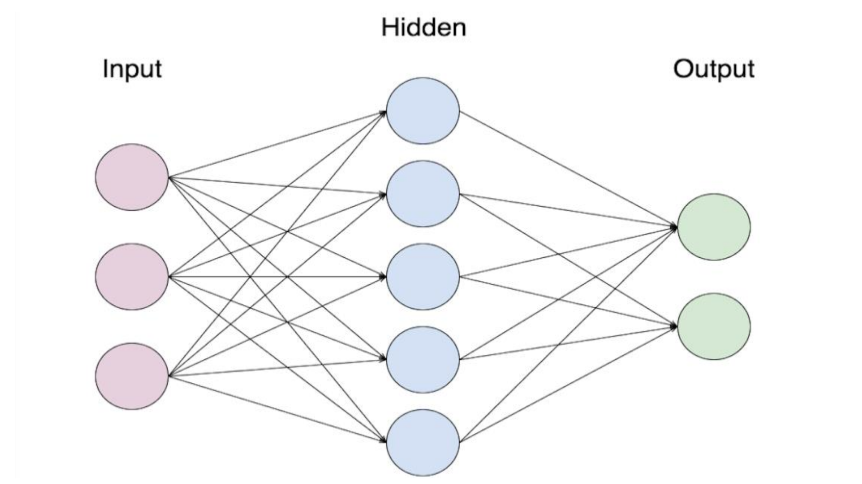
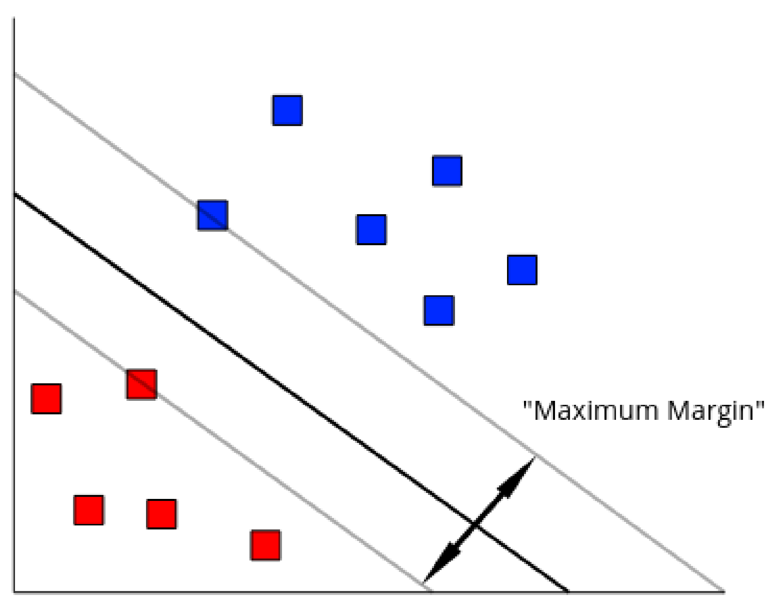
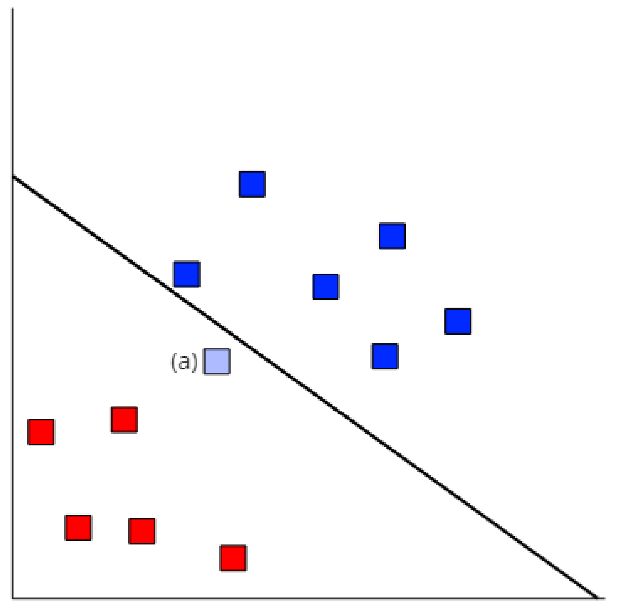


Figure 1: A simple neural network. (source: blog.webkid.io)

II. Simple Vector Machines

Simple Vector Machines (SVMs) are another tool in Machine Learning (ML) that attempt to learn a set of linear separators through a given dataset. SVMs are unique in that they have a maximum margin guarantee (Figure 2, 3). The "maximum margin" separator through a dataset, in general, is a good choice since it allows the most examples to fall within this "margin" and be correctly classified. The hyperplane shown is a decision boundary created by the SVM. Everything above it will be classified as blue, and everything below it as red.





(Left) Figure 2: A "Maximum Margin" SVM separator. (Right) Figure 3: The new point (a) is misclassified by a sub optimal, non-"maximum margin" hyperplane.

III. Decision Margins in Neural Networks

Now that the intuition behind margin maximization has been described, we may focus on designing a method to deliberately increase the decision margin for a neural network in a binary classification task. Currently, when training a network, there are no "margin guarantees": the decision boundary may be close to or far from all data points, and we have no way of knowing. We cannot simply "check" what the margin of a neural network is because it is a large system of nodes and connections, and not simply a function in a 2-dimensional space with a closed form.

One way to guarantee that the margin will increase in a neural network is to add new data points which the network will use to learn itself with. In the training phase of a network, the network attempts to classify every image in the training dataset correctly (Figure 4). We can exploit this to make it learn fake data points that purposely increase the distance between the hyperplane and all the true data points.

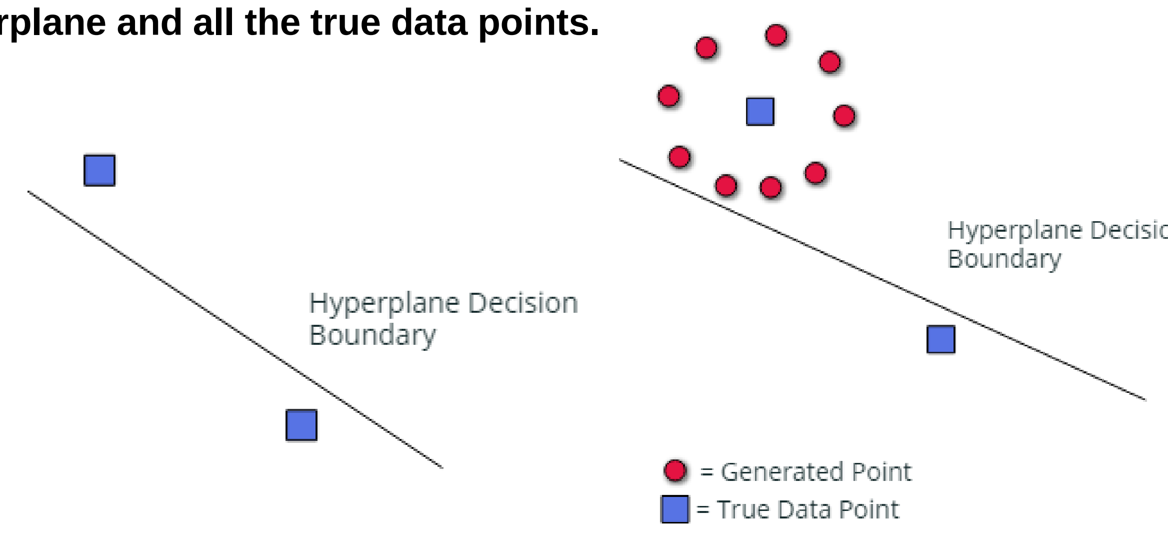


Figure 4: Slightly altering the decision boundary of a network by "pushing" it away from an existing datapoint.

IV. Methods

To "push" this decision boundary outward, and by proxy maximize the margin, we need to create a set of generated data points. In Figure 4, the datapoints lie in 2-dimensional space. However, if we are dealing with images, they may reside in much higher dimensional space. For example, a 28x28 pixel image resides in 784-dimensional space—something much more difficult to visualize on a piece of paper.

A simple way to construct these generated data points is to uniformly sample from an *n-sphere* around each true data point (*n* being the dimensionality of our input). For example, we operate on a dataset of 10,000 cats and dogs. For each image *m* in this dataset, we uniformly select 10 images from a 784-sphere around *m*, since each image is 28x28 pixels.

How do we choose the radius, *r*, for this sphere? We start with: *r*::=half the *minimum* distance between data points of opposing classes (Figure 5).

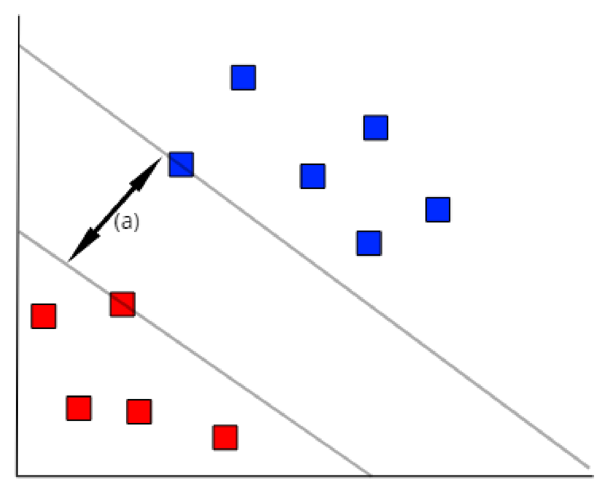


Figure 5: If (a) corresponds to the minimum distance between a pair of points from opposing classes, r=half this distance.

V. Results

As a control set, we teach a separate network using only native, unaugmented data, with a dataset of 10,000 cat and dog images. We are able to achieve ~73% accuracy on a set of data we withhold from the network during training, called the "validation set". Results on the validation set truly reflect how well a network is performing, because the network has not seen any images from it before.

We then generate our 20 random images per image in our original dataset, at a radius value of r=320 (a value computed using methods in [IV]). The more images we generate per original image, the greater probability guarantee of learning an expanded margin. We binary search for a margin that seems to work, testing on different *r* values (Figure 5).

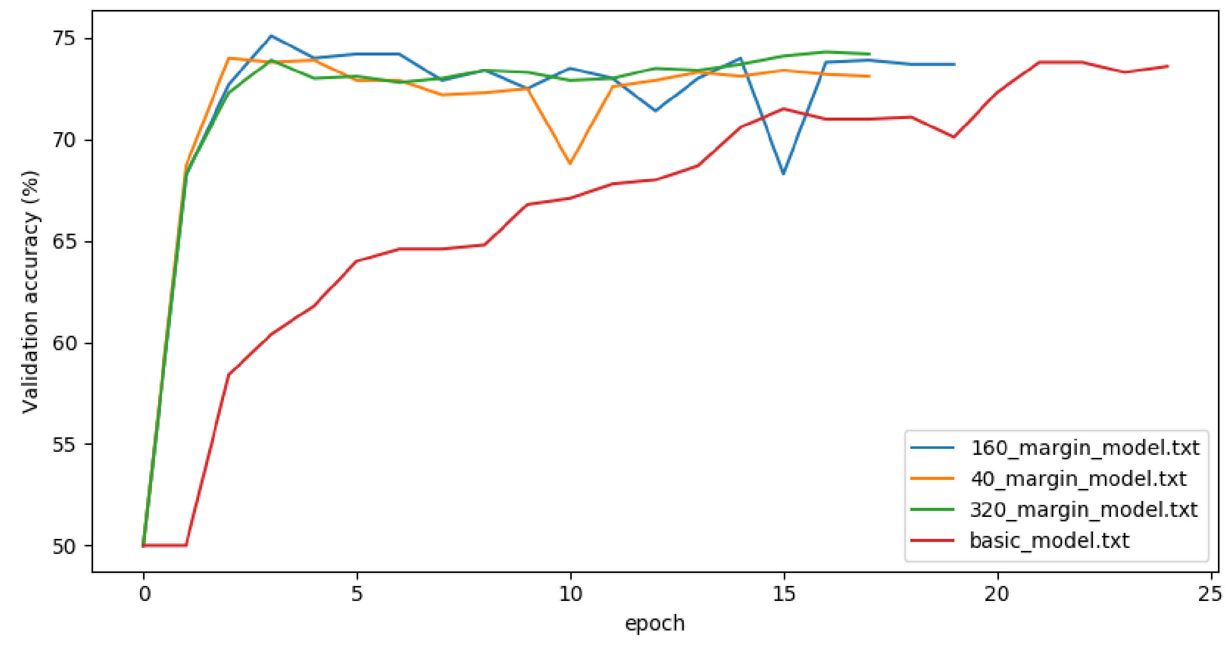


Figure 5: Accuracy of network trained on various margins. basic_model is trained on 0-radius, un-augmented data and is therefore a baseline.

The results indicate that this method of attempting to improve generalization is not effective. Although there is no way to compute the true "uncertainty" of our results and to measure whether our result is truly improving or degrading final accuracy, we can estimate this uncertainty to be around 2-3%. That is, if we were to increase our accuracy by 4% or more over a 10 run average, this would hold as a "significant result".

VI. "The Curse of Dimensionality" and Future Work

The key idea that we failed to take into account when doing these experiments is what is called the "Curse of Dimensionality". As we increase our spatial dimension, i.e. from 2→784, datapoints become more uniformly equidistant throughout space. Although we may be explicitly increasing the margin between *existing* datapoints with this method, it doesn't imply that any new, unseen point will fall into the same feature space we have augmented. Our increased margin therefore is not put to any use, and at worse we do not *decrease* the effectiveness of our classifier.

We are currently working on implementing the same idea after each layer in a convolutional neural network, which has dimensionality reduction embedded between layers.

VII. Acknowledgements

I would like to thank Dr. Ruozzi, who graciously takes the time to meet with me each week, and points me in new directions with each failure.

Experiments are open source: github.com/sid-devic/deep-margins.