ECE472 – Quiz 3

Jonathan Lam

September 22, 2020

1. *Describe the main method and key takeaways from "probes." What was the main hang-up that prevented the authors from completing more significant experimentation?* (arXiv:1610.01644)

   The authors attempt to make the effects of layers of a neural network more interpretable. Their method estimates how well any particular layer estimates the result; i.e., if a softmax function (in the case of a classification problem) with optimized parameters were applied on the inputs of that layer, and evaluated with the typical benchmarks (e.g., loss or classification error rate). In other words, it is as if we treat this layer as the first layer in the neural network, and only have a single linear classification layer to estimate the output. This involves its own optimization (for each layer) that must be independent of the main training optimization (which can be achieved by calculating gradients separately or manually stopping gradient backpropagation) – hence the name "probe."

   Using this method, the authors discover that the error with these linear classifier probes is strictly decreasing with increasing depth (from the input) in an optimized neural network; in other words, each layer can be thought of as acting as a better linear classifier than the previous; or in other other words, each layer makes the features more linearly-separable. This shows the (perhaps not obvious) effect that no layer in an optimized network can increase the error, which the authors term the "'greedy' aspect" of neural networks. By probing different layers, one can view how much a particular layer decreases the linear classification error, which can be used in diagnosing or removing unnecessary layers from a neural network.

   While the authors seemed to have great success with their methods, they had problems when layers had a very large width. Normally, very wide layers are manageable because all of the data in that layer is part of a function "pipeline" and can immediately be processed by the next layer and immediately discarded. However, in this method we are interpreting (and thus storing data about) the inputs of every layer we want to study. The authors state that when there a layer is millions of inputs wide, this may be on the magnitude of gigabytes (for that layer alone). They state a few ways to mitigate this error (mostly involving reducing dimensionality).

2. *Describe the main method and key takeaways from "confidence penalty."* (arXiv:1701.06548)

   As a motivation for their work, the authors state that a functional view of machine learning is about how good the distribution of output probabilities is, given some input. This can be thought of as how well a model generalizes (hence relating to regularization) but does not care about the internal parameterization of a neural network like the other forms of regularization do. Thus, the authors study the effect of penalizing models based on the (entropy of the)

distribution of their outputs; in other words, they look for models that has a "smoother" output distribution. Similar methods of "output regularization" include smoothing or adding noise to labels, label dropout, and distillation.

This "confidence penalty" is performed by adding an entropy term to the loss function (negative log-likelihood of the softmax (multiclass-classification) function) to penalize models with a lower entropy. (Additionally, an annealed confidence penalty and a hinge loss function help aid convergence rates.) In general, they find that this model outperforms the aforementioned other methods of "output regularization" in almost all of their test cases.

3. *Is there a difference at inference time between batch-norm and batch-renorm?* (arXiv:1702.03275)

   No. Batch-renorm attempts to rectify (minimize) the difference between the normalization between training and inference that occurs in batch-norm (i.e., training uses minibatch mean and standard deviation, while inference uses population/running average mean and standard deviation). In particular, this is done by altering the *training* stage by applying an affine transformation to the normalization step to make it more closely approximate the average mean and standard deviations (as used by the inference), and thus batch-renorm does not change how inference is carried out.