INFO 7375 - Neural Networks & AI

Homework to Chapter - 9

Submitted By:

Abdul Haseeb Khan
NUID: 002844724
khan.abdulh@northeastern.edu

To which values to initialize parameters (W, b) in deep neural networks and why?

Initialize weights with Xavier initialization for each layer and set biases to zero, because keeping activations zero mean and preserving their variance across layers prevents exploding or vanishing signals and supports effective convergence during training.
A recommended choice is Xavier initialization or one of its derived methods, where all weights in layer s are drawn from a normal distribution with mean μ=0 and variance $\sigma^2 = \frac{1}{n[s-1]}$, while biases are initialized with zeros, which helps maintain zero mean and stable variance of activations across layers so gradient signals do not explode or vanish.

Describe the problem of exploding and vanishing gradients?

In a deep network, backpropagated gradients can be amplified or minimized as one moves from the output layer toward the input layer, which can be seen by considering linear activations where $a = a[L] = W[L]W[L-1]\cdots WX$ so repeated matrix products scale signals layer by layer.
If every weight is initialized slightly larger than the identity, the activations increase exponentially with layer index, for example simplifying to $a = W[L]1.5^{L-1}x$, which produces exploding gradients during backpropagation so gradients become too big and the cost oscillates around its minimum.
Conversely, if every weight is initialized slightly smaller than the identity, the activations decrease exponentially, for example simplifying to $a = W[L]0.5^{L-1}x$, which produces vanishing gradients so gradients become too small and the cost seems to converge before reaching the true minimum.
All in all, initializing weights with inappropriate values leads to divergence or a slow down in training, and the observation generalizes beyond symmetric matrices to any initialization that is too small or too large, which is why choosing values that enforce zero mean and preserved variance across layers is emphasized.

Describe various types of initialization.

The various types of initializations are:

Constant initialization means that initially all weights and biases are set to a constant value, with the simplest form being zero initialization. For neurons with ReLU activation, the bias can be set to a small positive value like 0.1 or less so the gradient is likely nonzero at initialization, avoiding the dying ReLU problem.

Random initialization is listed among the practical techniques but no further descriptive details are provided in the material beyond its inclusion in the list.

Xavier (Glorot) initialization
The recommended initialization is Xavier for every layer s, which helps maintain stable signal propagation. Weights are picked randomly from a normal distribution with mean μ=0 and variance σ2=1/n[s−1] so that $W^{[s]} \sim N(\mu = 0, \sigma = \frac{1}{n^{[s-1]}})$, and biases are initialized with zeros. This aligns with rules of thumb that the mean of activations should be zero and the variance should stay the same across every layer. Under these assumptions, the backpropagated gradient signal should not be multiplied by values that are too small or too large, allowing it to travel to the input layer without exploding or vanishing.

Describe training, validation, and testing data sets and explain their role and why all they are needed.

Training, validation, and testing are three distinct subsets of a representative dataset, where the training set teaches the model by fitting parameters, the validation set tunes and evaluates the model and its hyperparameters during development, and the test set provides an unbiased final assessment on completely unseen data with no further adjustments. All three are needed to organize the empirical training process, optimize architecture and hyperparameters without contaminating evaluation, and measure how well the learned model generalizes to new data in a fair and final way.

Training data are collections of examples used to teach or train the model so it can understand patterns and relationships and learn to make predictions by adjusting the weightings between neuron connections based on feedback from performance during training. Validation data contain different samples used during the training phase to assess performance and fine tune the model's parameters, hyperparameters, and architecture in an iterative process that reveals how well the model is learning and adapting before it is finally put to the test. The test set is used after the model has been fully trained to assess performance on completely unseen data as an unbiased final evaluation, mirroring real world data the model has never seen, and during this final evaluation there is no adjustment of hyperparameters. In practice, activities align with these roles: training adjusts ANN parameters, validation tunes ANN hyperparameters and parameters, and testing is final testing on high quality data with no adjustment or tuning in this stage.

Separating these subsets enables a model that generalizes well to new data, with the test set serving as a proxy for new data and helping verify that a simple model that does not perfectly fit training data can still perform similarly on test data rather than overfit. Keeping validation and test sets separate is essential because otherwise an overfit might occur, and proper cross validation procedures and restraint in overemphasizing validation metrics help minimize overfitting and avoid discovering spurious relationships that do not apply in the real world. Never training on test data is critical, because surprisingly good evaluation results can indicate accidental leakage, such as duplicates between train and test, which destroys an unbiased final evaluation and misrepresents generalization. Validation and test sets must come from the same distribution while training data can come from various sources, which ensures that evaluation reflects the deployment distribution and that quality remains the major focus for verification and test while training may aggregate from multiple sources to increase the amount of samples. Reasonable initial splits such as 80 percent train, 10 percent validate, and 10 percent test are suggested, with alternative ratios depending on dataset size, for example 70–15–15 for 100 to 1,000 samples, 90–5–5 for 10,000 to 100,000 samples, and 98–1–1 for over 1,000,000 samples to keep substantial training data and reasonable validation and test counts. A validation dataset must not be too small to avoid an untuned or inaccurate model, and training typically takes multiple epochs, where an epoch is one training cycle through a complete dataset, with the appropriate train–validate–test ratio depending on the use case and improved through experience.

What is training epoch?

Epoch in artificial neural networks is one training cycle through a complete dataset. One cycle through a complete dataset in artificial neural networks is called an epoch and training a model usually takes more than one epoch.

How to distribute training, validation, and testing sets?

The optimum ratio when dividing records among train, validate, and test depends on application usage, model type, and data dimensions. Generally, apportioning 80% of the data to train, 10% to validate, and 10% to test is a reasonable initial split. With a typical dataset size in the range from 100 to 10,000 samples, the split can be around 70-15-15 or 80-10-10 to keep a substantial amount for training and reasonable numbers for validation and test. For large datasets from 10,000 to 100,000 samples the proportions may change to about 90-5-5, and for very large datasets over 1,000,000 samples the proportions may change even more to about 98-1-1. Validation and test sets must come from the same distribution while the training set may come from various sources, and verification and test data should come from the same source with a focus on quality. Training data may be appended from different sources with a focus on the amount of samples, and a validation dataset must not be too small or the model will be untuned and metrics like the F1 score will vary too widely. Never train on test data, since surprisingly good evaluation results can indicate that test data leaked into training, as illustrated by a case where an 80-10-10 split produced 99% precision on both training and test due to duplicate examples.

What is data augmentation and why may it needed?

Data augmentation refers to generating new training examples for a dataset by adding minor alterations to data or using models to create new points in the latent space of the original data.
It is a strategy that, though not strictly a regularization method, has its place, and can also be seen as a form of noise injection in the training dataset. Augmented data are derived from original images with minor geometric transformations such as flipping, translation, rotation, or the addition of noise to increase diversity, which differs from

synthetic data that are generated without real-world images, often via Generative Adversarial Networks.

Data scarcity is a common challenge in building deep learning models, and collecting such data can be costly and time consuming, so leveraging data augmentation helps reduce dependency on collection and preparation and build high precision AI models quicker. Large annotated datasets play a critical role because deep learning models need a lot of data to be trained, yet annotation can be difficult and expensive, therefore proper data augmentation is useful to increase model performance. More training data means lower model's variance, also known as lower generalization error, which directly motivates augmenting the dataset.