Projecting 3D Gaussian Data onto a Unit Sphere Using Deep Learning

Noppachon Chaisongkhram
School of Engineering and Computer Science, Victoria University of Wellington
chaisonopp@myvuw.ac.nz

***Abstract*—This report addresses a regression task of projecting 3D Gaussian-distributed data points onto a unit sphere using deep learning. A feed-forward neural network with three hidden layers was trained using mean squared error (MSE) loss and stochastic gradient descent. The model achieved a mean cosine similarity of 0.9994 and a mean radial error of 0.0288 radians (1.65°) on test data, demonstrating high accuracy in directional preservation. Code is publicly accessible via GitHub and Google Colab.**

***Index Terms*—deep learning, regression, unit sphere, PyTorch, 3D projection.**

I. Introduction

This report details a neural network solution for projecting 3D Gaussian-distributed points onto a unit sphere. The task involves regressing input points $\mathbf{x} \sim \mathcal{N}(0, I)$ to outputs $\mathbf{y} = \frac{\mathbf{x}}{\|\mathbf{x}\|}$ on the sphere. A feed forward network was trained using mean squared error (MSE) loss with explicit gradient descent. The solution demonstrates how deep learning can approximate geometrically precise transformations.

II. Theory

The spherical projection of a 3D point $\mathbf{x} = (x, y, z)$ is mathematically defined as:

$$\mathbf{y} = \frac{\mathbf{x}}{\|\mathbf{x}\|}, \quad \|\mathbf{x}\| = \sqrt{x^2 + y^2 + z^2} \tag{1}$$

where $\|\mathbf{x}\|$ denotes the Euclidean norm [1, 2]. This operation preserves the direction of $\mathbf{x}$ while scaling it to unit length. For the singularity at $\mathbf{x} = \mathbf{O}$ (a probability-zero event), the norm is exactly zero only if all three components are simultaneously zero. Since each component is a continuous random variable, the probability of any single component being exactly zero is zero. Thus, allowing for no safeguards against this event.

The neural network $f_\theta(\mathbf{x}))$ acts as a universal approximator of the true projection function $f^*(\mathbf{x}) = \mathbf{y} = \frac{\mathbf{x}}{\|\mathbf{x}\|}$. Its architecture comprises an input layer $\mathbb{R}^3$ with LeakyReLU activation ($\alpha = 0.01$), two hidden layers (each with 20 units and LeakyReLU activation), and a linear output layer $\mathbb{R}^3$. The hidden layers follow the transformation:

$$\mathbf{z}^{(i)} = g(\mathbf{W}^{(i)}\mathbf{z}^{(i-1)} + \mathbf{b}^{(i)}), \quad g(\cdot) = \text{LeakyReLU}(\cdot) \tag{2}$$

where g is the LeakyReLU nonlinearity [3]. Training employs the mean squared error (MSE) loss [4,5]:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \left\| \mathbf{y}^{(i)} - f_\theta(\mathbf{x}^{(i)}) \right\|_2^2 \tag{3}$$

Under the probabilistic assumption of isotropic Gaussian noise in the outputs, minimising MSE corresponds to maximum likelihood estimation. Optimisation uses the Adam algorithm, with gradients computed explicitly via back propagation.

III. Experiments

The experimental setup began with generating 5000 samples from a 3D standard normal distribution. Data were partitioned into training (70%), validation (15%), and test (15%) sets. The network was initialised using Xavier uniform initialisation and trained with Adam (learning rate 0.001) and a batch size of 64. Early stopping (patience 10) monitored validation loss to prevent over-fitting.

Performance was quantified using two metrics: angular error ($\cos^{-1}(\mathbf{y}_{\text{pred}} \cdot \mathbf{y}_{\text{true}})$ in degrees) [6], and cosine similarity ($\frac{\mathbf{y}_{\text{pred}} \cdot \mathbf{y}_{\text{true}}}{\|\mathbf{y}_{\text{pred}}\| \|\mathbf{y}_{\text{true}}\|}$) [1,7,8]. Results confirmed robust learning, both training and validation MSE decreased monotonically, converging after 178 epochs. Test performance achieved an MSE of 0.000721, mean angular error of 1.65°, and and near-perfect cosine similarity of 0.999456. Visualisations further verified that predictions aligned closely with ground truth positions on the sphere.

IV. Conclusion

The neural network successfully learned the spherical projection mapping, achieving remarkably high directional accuracy with a mean angular error of 1.65°. This performance demonstrates the network's ability to capture the geometric relationship between 3D Gaussian inputs and their spherical projections with near-perfect precision.

V. Statement

This work represents my original implementation. The solution was developed using PyTorch, NumPy, and Matplotlib within PyCharm Jupyter Notebook. The complete code, including data generation and visualisation routines, is accessible via GitHub and Google Colab at:

- https://github.com/jescsk/AIML425/tree/main/Assignment1
- https://colab.research.google.com/drive/1cG8uxKYtN16x57ll4-BGlgSBLv5tde7G?usp=sharing

REFERENCES

[1] G. B. Arfken, H. J. Weber, and F. E. Harris, Mathematical methods for physicists a comprehensive guide. Amsterdam (Holanda) Elsevier, 2013.

[2] Erwin Kreyszig, Advanced Engineering Mathematics. Editora: S.L.: Wiley, 2020.

[3] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, Massachusetts: The MIT Press, 2016. Available: https://www.deeplearningbook.org/

[4] K. P. Murphy, Machine learning : a probabilistic perspective. Cambridge (Ma): Mit Press, 2012.

[5] C. Bishop, Pattern recognition and machine learning. Springer Verlag, 2006.

[6] A. Hanson, Visualizing quaternions. San Francisco, Ca: Morgan Kaufmann ; Amsterdam ; Boston, 2006.

[7] Horn RA, Johnson CR. Matrix Analysis. Cambridge University Press; 1985.

[8] Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge: Cambridge University Press.