Kendall Beaver
Prasanth Gubbala
Venkata Satya Murali Krishna Chittlu

Proposal

Project Goal

Our goal is to create a Bayesian Neural Network, with the help of the Monte Carlo Dropout regularization technique, that can accurately predict the stress level of any employee across the world, where stress level is defined as *low, medium,* and *high*, and the employee as working in a role that is *remote*, *hybrid*, or *onsite*. So given metadata about an employee—their gender, age, health, job and industry, how many hours they work, etc.—our model will predict a single value as to what their stress level will most likely be.

Project Questions

1. How effectively can a neural network predict employee stress levels worldwide using specific individual and career-related metadata?
2. How does the dropout technique impact the performance and optimization of a neural network in predicting stress levels?

Dataset

We propose using a 2024 stratified dataset from Kaggle, Remote Work & Mental Health, that equally captures 5,000 samples across various heterogeneous groups (age, gender, career, country, etc.), and we intend on using all 19 variables from the dataset because stress is a multifaceted condition that's the result many, different lifestyle and career factors—not just one or a few factors. The variables in the dataset are:

1. Age
2. Gender
3. Job Role
4. Industry
5. Years of Experience
6. Work Location (***response variable***)
7. Hours Worked Per Week
8. Number of Virtual Meetings
9. Work Life Balance Rating
10. Stress Level
11. Mental Health Condition
12. Access to Mental Health Resources
13. Productivity Change
14. Social Isolation Rating
15. Satisfaction with Remote Work
16. Company Support for Remote Work
17. Physical Activity
18. Sleep Quality
19. Region

Priors - Parameters & Hyperparameters

Since Gaussian distributions are smooth and differentiable and crucial for optimization in neural networks, we will use Gaussians as our priors using the parameters age, gender, industry, work location hours worked as main, we will explore the possibility of other parameters impact as a prior. And since we plan on using a basic, feedforward neural network with a few layers, we need to set the values (μ, σ) for these model parameters: the weights (W_{ij}) and biases (b_j).

We do not know what the mean (μ) values will be for the weights in each layer on our model, if they'll tend to take on positive or negative values, and have therefore chosen to set μ at 0. We do have more confidence that there will be less variance (σ) in the weights, that the model will always predict the right outcomes, and have set σ to 0.5.

$$W_{ij} \sim N\,(0,0.5)$$

Since biases seem to be kept near 0 in most basic, feedforward neural networks, we have chosen μ and σ to reflect this:

$$b_j \sim N\,(0,0.1)$$

These parameters are, of course, dependent upon the values that we set for hyperparameters being set correctly, which for our initial test we plan on using 2 hidden layers, 50 neurons per layer, a ReLU activation for the hidden layers, a softmax function for the output layer, and a dropout rate (p) of 0.3 since the model seems to overfit the data when p=0.1, and overfits the data when p=0.5.

Training & Coding in R

Since we have a well-sampled, stratified dataset, we believe that an 80/20 train-test split is appropriate to use, that it's not necessary to increase the test set beyond 20%. We may also supplement our training with a 5-, 10-, or 5 & 10-fold cross-validation, or may outright use k-fold cross-validation instead of a test-train split—to be determined.

And since we don't intend on building a complex, specialized neural network, the TensorFlow library in R should more than suffice for our needs in this project. In case there are instances where we need to code in Python, we will convert the code appropriately.