Deep Learning Course Project

Princy Gautam (B20BB051)

Predicting mRNA Abundance Directly from Genomic Sequence Using Deep Convolutional Neural Networks

Introduction

Protein types and concentrations have the greatest impact on cellular function. Previous research indicated that translational regulation plays an important role in determining protein abundance, implying that mRNA levels may not be as predictive of protein abundance. Reanalysis of the data revealed that mRNA levels can explain up to 84% of protein level variations, with transcription rates accounting for 73% and mRNA degradation rates contributing 11%. This finding demonstrates that protein abundances can be predicted with a high degree of accuracy based on mRNA levels.

To what extent is gene expression predictable directly from the genome sequence?

The transcriptional activities of separated promoters can explain about 54% of the differences in the activity of endogenous promoters. This result shows that there is a physical link between the sequence of promoters and the different levels of gene expression. So, it suggests that it might be possible to make a learnable mathematical function with the right parameters and could correctly predict the levels of mRNA expression based only on genomic sequence information.

Motivation

The fundamental challenge in biology is to understand the complex relationship between genetic information and gene expression. The ability to predict mRNA abundance solely from the promoter sequence holds immense potential for unraveling the mechanisms governing gene regulation. Traditional methods of studying gene expression are costly and time - consuming and limited by their inability to capture the specific regulatory networks within the genome. By using the power of deep learning, the aim is to overcome these limitations and provide a more efficient and accurate approach to predict mRNA abundance.

Model architecture and Codeflow

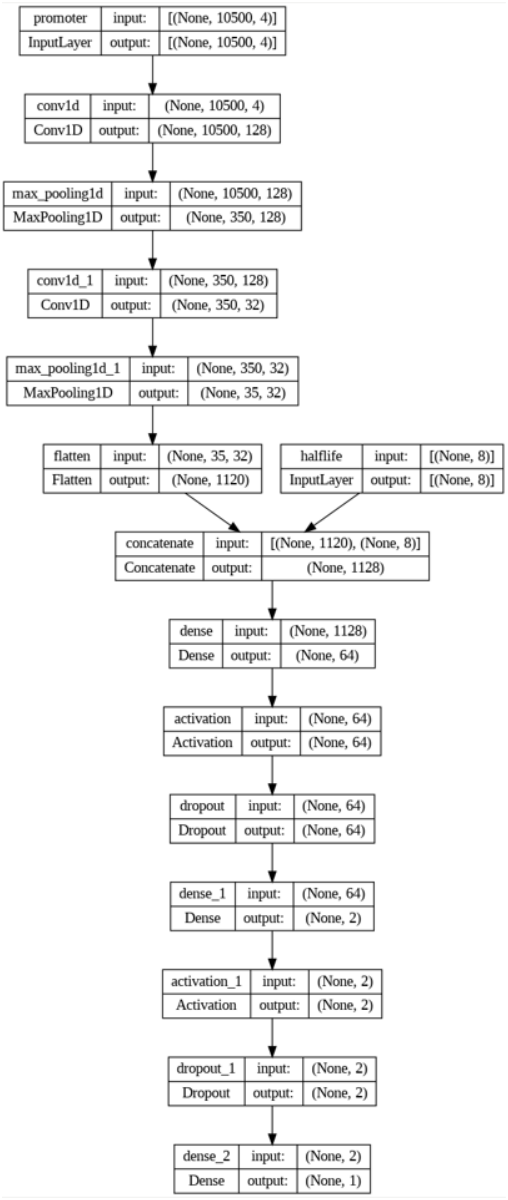
The code is a script for training and using the Xpresso model to predict median gene expression levels in humans. Overview of the code is as follows:

- Downloading data:
    - It uses the ‘wget’ command to download three datasets related to predicting gene expression levels: human training, human validation, and human testing datasets.
    - It also downloads a ZIP file containing input sequences for generating predictions.
- Installing dependencies:
    - It uses pip to install two Python packages: Biopython and hyperopt.

    - These packages provide additional functionality required by the code.
- Importing libraries:
    - It imports various libraries required for the code, including TensorFlow, Keras, NumPy, pandas, and others.
- Setting up GPU device:
    - It checks if a GPU device is available and raises an error if not found.
- Defining functions and variables:
    - It defines several functions and initializes global variables used throughout the code.
    - **Function 1:** objective function - the objective function performs the training and evaluation of a deep learning model with specific hyperparameters, aiming to minimize the mean squared error loss on the validation set.
    - **Function 2:** one_hot function - function takes a sequence of nucleotides as input and converts it into a one-hot encoded representation using a NumPy array.
    - **Function 3**: generate_predictions - uses a trained model to generate predictions for input sequences and saves the predictions to an output file.
    - **Function 4:** main function - is the entry point of the code and is responsible for setting up the data and parameters, running the optimization process, and printing the results.
- Defining hyperparameters:
    - It defines a dictionary 'params' that contains hyperparameters for the Xpresso model.
    - The hyperparameters are defined using the hyperopt library and specify various settings for the model architecture.
- Defining the objective function:
    - It defines the objective function that takes the hyperparameters as input and trains the Xpresso CNN model using those parameters.
    - The function constructs the model architecture based on the hyperparameters, compiles it with an optimizer and loss function, and trains it on the training data.
    - It also evaluates the model on the validation data and calculates the mean squared error (MSE).
    - Finally, it evaluates the performance of the best model on the test data, calculates the R-squared value, and saves the predictions to a file.
- Training the Xpresso model:
    - It calls the main function, which sets the data directories, loads the training, validation, and test data, and calls the objective function to train and evaluate the model.

Results

Our model architecture is as follows:



The predictions generated on a dataset of 1000 promoters with mean half-life is observed as follows:

predictions.txt

```
Gene  Pred  Actual
b'ENSG00000102547'  b'0.005055092'  b'0.18310068655135697'
b'ENSG00000254901'  b'0.005055092'  b'0.47618292960685443'
b'ENSG00000138380'  b'0.005055092'  b'-0.1387473122816179'
b'ENSG00000103489'  b'0.005055092'  b'0.4328246299405266'
```

The deep learning approach leverages the ability of CNNs to automatically learn hierarchical representations from input data, enabling the model to capture meaningful patterns and make predictions based on promoter sequences and mRNA half-life data.

Key DL approach

The deep learning approach used in the provided code is based on a Convolutional Neural Network (CNN). CNNs are a type of deep learning model commonly used for processing and analyzing visual data, but they can also be applied to sequence data, such as DNA sequences.
In the code, the CNN is employed to learn patterns and features in the promoter sequences that are relevant for predicting gene expression levels. The CNN architecture consists of convolutional layers, pooling layers, and fully connected layers.

Here is a high-level overview of the deep learning approach:

- Input Data:
    - mRNA Half-life Data (halflifedata): Represents the half-life values of mRNA molecules.
    - Promoter Sequences (input_promoter): Represents the DNA sequences that control gene expression.
- Convolutional Layers:
    - The input promoter sequences are passed through one or more convolutional layers.
    - Convolutional filters scan the input sequences to capture local patterns and features.
    - The filters learn to detect relevant motifs and sequence characteristics.
- Pooling Layers:
    - After each convolutional layer, pooling layers are applied.
    - MaxPooling1D layers are commonly used in sequence analysis tasks.
    - Pooling reduces the spatial dimensions of the output, extracting the most salient features.
- Flatten Layer:
    - The output of the last pooling layer is flattened into a 1-dimensional vector.
    - This allows the subsequent fully connected layers to process the learned features.
- Concatenate Layer:
    - The flattened output from the pooling layers is concatenated with the halflife data.
    - This combination of sequence information and halflife data is used to make predictions.
- Fully Connected Layers:
    - The concatenated features are fed into one or more fully connected (dense) layers.
    - These layers learn complex relationships between the input features and the target gene expression levels.
- Activation and Dropout Layers:
    - Activation functions introduce non-linearity, allowing the model to learn complex mappings.

    - Dropout layers randomly deactivate a fraction of neurons during training to prevent overfitting.
- Output Layer:
    - The final fully connected layer produces a single output, representing the predicted gene expression level.
- Training and Optimization:
    - The model is trained using the mean squared error loss function.
    - Stochastic Gradient Descent (SGD) optimization is used to update the model's parameters.
    - The hyperparameters are tuned using a hyperparameter optimization library (e.g., hyperopt).

References

https://www.sciencedirect.com/science/article/pii/S2211124720306161
https://egg2.wustl.edu/roadmap/data/byDataType/rna/expression/57epigenomes.RPKM.pc.gz
https://www.semanticscholar.org/paper/A-deep-auto-encoder-model-for-gene-expression-Xie-Wen/7b0832a59aa2dd6059665cd1062928444ab29f07
https://github.com/vagarwal87/Xpresso
https://www.semanticscholar.org/paper/A-new-LSTM-based-gene-expression-prediction-model%3A-Wang-Li/97dcdc4186c3eab15042ed942376bde2cf2a1847