Linear Models: Still the Best Thing Since Sliced Bread (I need a real title)

This manuscript (permalink) was automatically generated from greenelab/linear_models_manuscript@9b17436 on December 6, 2021.

Authors

- **Benjamin Heil**
  0000-0002-2811-1031 · ben-heil · autobencoder
  Genomics and Computational Biology Graduate Group, Perelman School of Medicine, University of Pennsylvania ·

Funded by Grant XXXXXXXX

Abstract

Methods

Data

Our analyses used bulk RNA-seq data downloaded from the recount3 compendium [CITE] on TODO date. Before filtering, the dataset contained 317,258 samples, each containing 63,856 genes.

We then preprocessed the data. To filter out single-cell data, we removed all samples with a sparsity greater than 75 percent. We also removed all samples marked 'scrna-seq' by recount3's pattern matching method (stored in the metadata as 'recount_pred.pattern.predict.type').

To ensure the samples were comparable, we converted the data to transcripts per million [CITE] using gene lengths from BioMart [CITE]. To ensure the genes' magnitudes were comparable, we performed standardization to scale each gene's range from zero to one. We kept the 5,000 most variable genes within the dataset.

Samples were labeled with their corresponding tissues using the 'recount_pred.curated.tissue' field in the recount3 metadata. These labels were based on TODO A total of TODO samples in our dataset had corresponding tissue labels.

Samples were also labeled with their corresponding sex using labels from Flynn et al. [CITE]. These labels were derived using pattern matching on metadata from the European Nucleotide Archive [CITE]. A total of TODO samples in our dataset had sex labels.

Data splitting

In our analyses we use five-fold cross-validation with two types of data splitting. The first type is samplewise splitting. In the samplewise paradigm, gene expression samples are split into cross-validation folds at random without respect to which studies they belong to. In the studywise paradigm, entire studies are assigned to cross-validation folds.

While samplewise splitting is common in the machine learning and computational biology literature [CITE?], it is ill-suited to gene expression data. There are study-specific signals in the data, and having samples from the same study in the training and validation sets causes information leakage [CITE https://www.nature.com/articles/s41576-021-00434-9 ?]. As a result, samplewise splitting inflates the estimated performance of the models. Studywise splitting avoids leakage by ensuring all the study-specific signals stay within either the training or the validation sets.

We use studywise splitting for the results in the main section of the manuscript, but have added the samplewise results to the supplement to show that the results are not an artifact of data splitting.

Model architectures

We use four representative models to demonstrate the performance profiles of different model classes.

Our first two models are fully connected neural networks implemented in PyTorch [CITE]. The first is a three layer network with hidden layers of size 2500 and 1250. Our second is a five layer network, with

hidden layers of size 2500, 2500, 2500, and 1250. Both models use dropout [CITE] with a probability of 0.5, and ReLU nonlinearities [CITE]. The deeper model uses batch normalization [CITE] to mitigate vanishing gradients.

The other two models are two implementations of linear regression. One model, 'sklearnLR', is a wrapper around scikit-learn's [CITE] implementation of logistic regression, while the other is a PyTorch implementation.

TODO either justify why each model was used here, or more likely put it in the results section.

Model training

Optimization

Our PyTorch models minimized the cross-entropy loss using an Adam [CITE] optimizer on minibatches of data. Our Scikit-Learn model used the LBFGS optimization algorithm on the entire dataset. Both model types used inverse frequency weighting to avoid giving more weight to more common classes.

Regularization

The PyTorch models used early stopping and gradient clipping to regularize their training.

Hyperparameters

The hyperparameters for each model can be found in their corresponding config file at https://github.com/greenelab/saged/tree/master/model_configs/supervised.

Determinism

Model trainining was set to be deterministic by setting the Python, NumPy, and PyTorch random seeds for each run, as well as setting the PyTorch backends to deterministic and disabling the benchmark mode.

Logging

Model training progress was tracked and recorded using Neptune [CITE].

Hardware

All analyses were performed on an Ubuntu 18.04 machine with 64 GB of RAM. The CPU used was an AMD Ryzen 7 3800xt processor with 8 cores, and the GPU used was an Nvidia RTX 3090. The pipeline can be run on a computer with lower specs, but would have to run fewer elements in parallel. From initiating data download to finishing all analyses and generating all figures, the full Snakemake [CITE] pipeline takes about TODO days to run.

Binary classification?

Workflow

Multitissue classification

The multitissue classification analyses were trained on the 21 tissues that had at least 10 studies in the dataset. TODO should I mention which tissues those are?

- Mention this formulation is used for both the tissue and sex prediction
- Which tissues and why
- How is the data split with respect to the tissues
- Metric for performance

Transfer learning

- Pretraining setup (train on different studies with same labels)
- Data splitting details

Pretraining

- Pretraining setup (imputation)
- How much data is used for pretraining?
- Matching initializations for pretrained and non-pretrained networks

Visualizing expression compendium

- Extracted 1000 expression samples at random
- Plotted via UMAP and PCA

References