Inference of CRISPRoff efficacy from gene features

Justin Lubin
SID [redacted]
Department of Computer Science
University of California, Berkeley
justinlubin@berkeley.edu

Colin Skinner
SID [redacted]
Department of Bioenginering
University of California, Berkeley
cmskinner2015@berkeley.edu

1 Introduction

The introduction of CRISPR molecular systems revolutionized the world of biology by enabling quick and easy modifications to the genome of an organism, with applications from agriculture to therapeutics. [1] However, CRISPR is difficult to reverse and toxic to cells, making it difficult to administer for human therapeutic purposes. [2]

CRISPRoff is a molecular system developed by Nuñez et al. in 2021 [3] that can suppress gene expression by modifying the molecules attached to the genome (known as the *epigenome*) rather than the genome itself, resulting in easier reversibility and less toxicity than other gene editing methods.

However, CRISPRoff does not work equally well on every gene, and biologists do not fully understand why. To aid in this understanding, we aimed to build interpretable machine learning models for inference of CRISPRoff efficacy from gene features.

2 Method

2.1 The dataset

We first describe how we operationalized the problem of inferring CRISPRoff efficacy from gene features into features of a dataset we built.

2.1.1 Operationalizing CRISPRoff efficacy

To operationalize CRISPRoff efficacy, we leverage an existing dataset produced by a wet lab experiment in which each gene is assigned a "phenotype score" (scalar value typically between -0.6 and +0.1) with a more negative score meaning CRISPRoff worked better at suppressing the gene expression experimentally. [3]

For example, Figure 1 shows a plot of the CRISPRoff scores against CpG island size, one of the gene features we investigated. Our goal is thus to predict these scores in an interpretable way from features of genes, which we discuss next.

2.1.2 Gene features

For the gene features, we aggregated and analyzed data from 10 different sources—including previous publications and wet lab experiments—for a total of 10 features. These features (many of which required a nontrivial amount of labor to analyze, clean, and combine) were:

1. `chromosome`: *Which chromosome the gene is on.* Computed from the Genome Reference Consortium's 38th version of the human genome (hg38). Data source: `https://www.ncbi.nlm.nih.gov/data-hub/genome/GCF_000001405.26/` (both DNA sequence and gene annotation files).
2. `gene_length`: *The length of the gene in DNA base pairs.* Computed from hg38, as above.

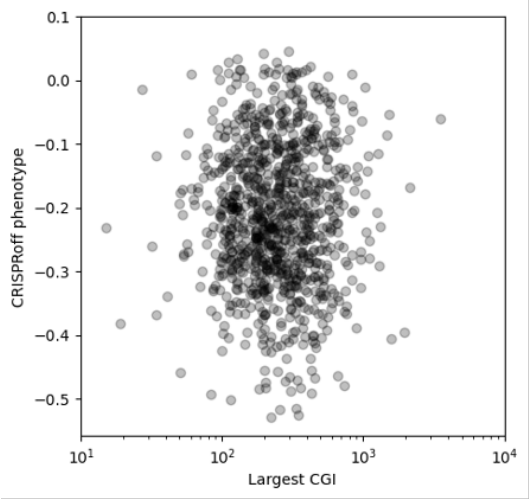

Figure 1: CRISPRoff phenotype scores vs. largest CpG island, one of the ten gene features we analyzed.

3. `nt_rep1_count` (non-targeting replicate 1 RNA count): *Average RNA expression of the gene*. Computed from an RNA sequencing analysis done on the same cell type as a control for the CRISPRoff experiment. Data source: `https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM5123398`.

4. `nt_rep2_count` (non-targeting replicate 2 RNA count): *A second replicate of average RNA expression of the gene, as above*.

5. `traditional_cgi`: *Whether the gene has a CpG island (CGI) according to the traditional definition [4]*. A CGI is a genomic region of length at least 200 base pairs with an unusually high density of the nucleotide cytosine (C) followed by the nucleotide guanine (G), called CpGs. DNA methylation refers to a particular type of epigenetic marker whereby a methyl group is chemically attached to a CpG site on the genome. Many, but not all, genes have CGIs. Data source: supplementary materials (Table S3) of [3].

6. `largest_cgi`: *The largest CpG-rich region of the gene*. This feature relaxes the definition of CGIs to allow for regions of any length with unusually high density of CpGs, and captures the length of the longest such region near the gene. Data source: custom algorithm over hg38 (dataset linked above).

7. `promoter_H3K4me3`: *The average amount of the epigenetic mark H3K4me3 present per base pair in the promoter region ($\pm$ 1500 base pairs of the start) of the gene.* This data is from a different wet lab experiment with the same cell type as the CRISPRoff experiment. Data source: `https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM2711412`

8. `promoter_H3K27ac`: *The average amount of the epigenetic mark H3K27ac present per base pair in the promoter region of the gene.* Similar to `promoter_H3K4me3` (same cell type, different experiment). Data source: `https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM2711409`

9. `promoter_H3K9me3`: *The average amount of the epigenetic mark H3K9me3 present per base pair in the promoter region of the gene.* Similar to `promoter_H3K4me3` (same cell type, different experiment). Data source: `https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM5123386`

10. `promoter_methylation` *The average amount of the DNA methylation (as described in the `traditional_cgi` feature) present per base pair in the promoter region of the gene.* Similar to `promoter_H3K4me3` (same cell type, different experiment). Data source: `https://www.encodeproject.org/experiments/ENCSR794HFF/`

These datasets use different names for the same gene, so we also used a mapping between different gene names from Ensembl.[1] Lastly, we used the Broad Institute's DepMap to get a list of essential genes,[2] which we discuss next.

2.1.3 An unfortunate caveat: essential genes

This CRISPRoff dataset we described in Section 2.1.1 provides scores for 20,360 human genes, but the experiment only produces meaningful scores (i.e., those that actually reflect how well CRISPRoff is working) for genes that are essential to cell life, which are called *essential* genes. After narrowing this list down to the essential genes and combining with the datasets we describe in the previous section, we were left with a total of 1,190 genes to analyze as sample points.

Our initial proposal indicated that we would work with 17,415 samples and 11 features. Based on the limitations of the wet lab experiment for producing the CRISPRoff scores, we later realized we needed to filter this down to 1,190 (essential) genes and 10 features; the feature we removed was whether the gene was essential or not.

2.2 The models

To perform inference, we chose three kinds of models that all perform some form of interpretable subset selection, the latter two of which go beyond our original project proposal:

- **LASSO regression** (linear regression with an L1 penalty),
- **LASSO classification** (logistic regression with an L1 penalty), and
- A shallow **decision tree** (with a maximum depth of 3).

To view our regression problem as a classification problem, we binarized the labels (CRISPRoff scores) with a threshold of -0.1; we considered CRISPRoff to fail (class 0) on scores above that threshold and to succeed (class 1) on scores below that threshold.[3]

For each of these models, we chose among different hyperparameters (discussed next) and data-preprocessing strategies (discussed in Section 2.3) by cross-validation with a 60/20/20 train/validation/test split (i.e., 5-fold cross-validation on 80% of the data). Our final models used the setup that minimized the cross-validation error (mean squared error for regression and classification error for classification). We used the test data **only** for the final evaluation in Table 1 and Figures 5 and 6, as discussed in Section 3.

Hyperparameters For the LASSO regression and LASSO classification, we used cross-validation to tune the L1 penalty parameter. For the decision tree, we used cross-validation to tune minimum impurity decrease at each split, which is essentially the required split entropy difference threshold.

2.3 Data pre-processing

We now discuss our different data pre-processing options. Each of the following categories are independent of the other categories, and we tried all combinations of selecting one choice from each category in our cross-validation setup.

Handling numeric variables For each of the numeric variables, we tried the following strategies:

NV1. Do nothing.

NV2. Center the data by subtracting the mean.

NV3. Standardization: center + normalize the data by subtracting the mean and dividing each feature by the feature's standard deviation.

[1] `https://www.ensembl.org/biomart/martview/4fb56a8d8b07d742189052a8f36c1756`

[2] `https://depmap.org/portal/download/all/?releasename=DepMap+Public+22Q4&filename=CRISPRInferredCommonEssentials.csv`

[3] This threshold was chosen before constructing any models via personal communication with the original experimentalist who produced the CRISRPoff phenotype score dataset, James Nuñez.

NV4. Min-max normalization: subtracting each feature entry by the minimum value of the feature and dividing by the maximum minus the minimum value.

Encoding gene expression We had two experimental replicates of a feature encoding gene expression in our design matrix. To handle this, we tried the following strategies:

GE1. Do nothing.

GE2. Make a new column for the average of the two replicates and a new column for their standard deviation. Drop the original two columns.

Encoding chromosomes One of our columns—which chromosome the gene resides on—was categorical with 24 different possible values (one for each human chromosome: `chr1`, `chr2`, ..., `chr22`, `chrX`, `chrY`). We tried the following strategies to encode these categorical variables:

C1. Make a one-hot encoding for each chromosome (24 new columns). Drop the original column.

C2. Make a column for whether the chromosome is a sex chromosome (`chrX` or `chrY`). Drop the original column.

C3. Make two new columns: one for whether the chrosome is `chrX` and one for whether the chromosome is `chrY`. Drop the original column.

Extra note: handling missing data Before restricting our attention to essential genes, we had samples with missing (continuous, numeric) data in our dataset. We implemented both dropping samples with missing data and imputing with the column mean for each missing data point. Fortunately, the essential genes had no missing data.

3 Results

As previously discussed, we tested all combinations of data pre-processing setups and hyperparameters mentioned above via cross-validation. Figures 2, 3, and 4 show the performance of our best models from cross-validation against their respective hyperparameters; our final models are those with hyperparameters that minimize the validation error in these graphs.

Table 1 shows the final evaluation of our best models from each of the three categories of model we looked at, including the hyperparameters, the pre-processing setup, the training error (when trained on both the 60% reserved for training during cross-validation as well as the 20% reserved for validation during cross-validation), and the test error (on the withheld 20% that was not used up to this point).

Additionally, Figures 5 and 6 show ROC curves for our two classifiers, which we obtained by varying the posterior thresholds from 0 to 1 in increments of 0.1.

Lastly, as all our models are interpretable, we can describe what each one has learned:

- **LASSO regression**. Nonzero feature weights:
  - `on_chr11`, 1.340×10^{-2}
  - `on_chr17`, 8.590×10^{-3}
  - `promoter_H3K27ac`, 7.577×10^{-3}
  - `nt_expression_std`, -7.762×10^{-6}
  - `nt_expression_level`, 6.319×10^{-6}
  - `largest_cgi`, -2.390×10^{-7}
  - `gene_length`, 3.645×10^{-8}
- **LASSO classification**. Nonzero feature weights:
  - `on_chr11`, -4.703×10^{-1}
  - `promoter_H3K27ac`, -3.514×10^{-1}
  - `on_chr1`, 2.975×10^{-1}
  - `traditional_cgi`, 2.867×10^{-1}

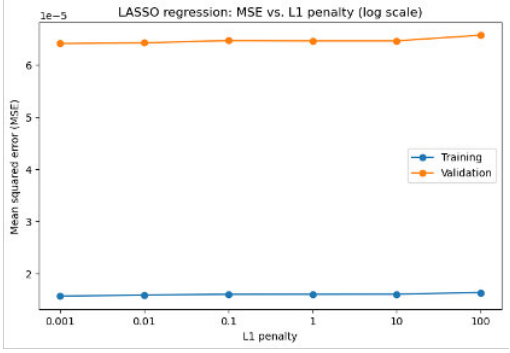

Figure 2: Training and validation mean squared error of our best LASSO regression model from cross-validation against possible L1 penalties (log scale). The minimum validation error occurs at 0.001.

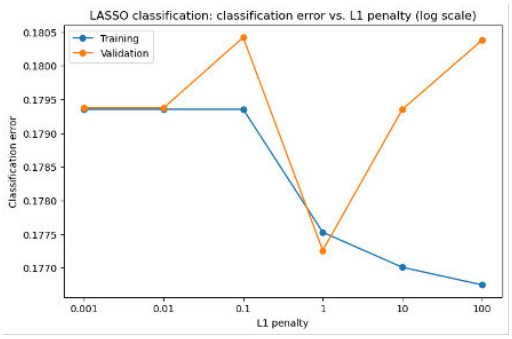

Figure 3: Training and validation classification error of our best LASSO classification model from cross-validation against possible L1 penalties (log scale). The minimum validation error occurs at 1.

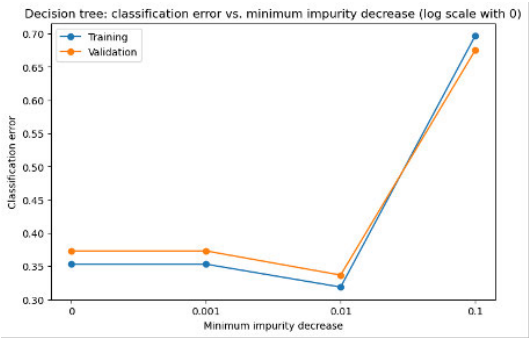

Figure 4: Training and validation classification error of our best decision tree model from cross-validation against possible minimum impurity decrease values (log scale with 0 added to the left). The minimum validation error occurs at 0.01.

  - `on_chr18`, 2.856×10^{-1}
  - `on_chr6`, 2.662×10^{-1}
  - `promoter_methylation`, 2.343×10^{-1}
  - `largest_cgi`, -1.716×10^{-1}
  - `on_chr17`, -1.508×10^{-1}
  - `nt_rep2_count`, -1.203×10^{-1}
  - `on_chr14`, -1.102×10^{-1}
  - `on_chr12`, 1.051×10^{-1}
  - `on_chr19`, -9.805×10^{-2}
  - `promoter_H3K9me3`, -4.366×10^{-2}
  - `gene_length`, 1.434×10^{-2}
  - `on_chr5`, -1.423×10^{-3}

- **Decision tree**. One split: is the centered average gene expression less than -952.265? If so, return "CRISPRoff succeeds" (posterior probability 57.95%). If not, return "CRISPRoff fails" (posterior probability 61.33%).

4 Discussion

We conclude by interpreting and contextualizing our results from the previous section.

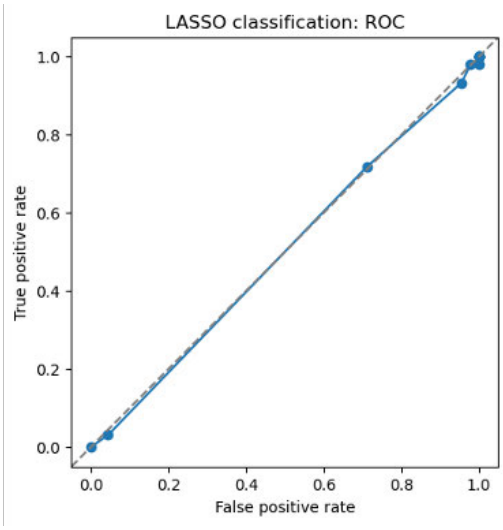

Figure 5: The ROC curve for our best LASSO classifier. We obtained this plot by varying the posterior threshold from 0 to 1 in increments of 0.1.

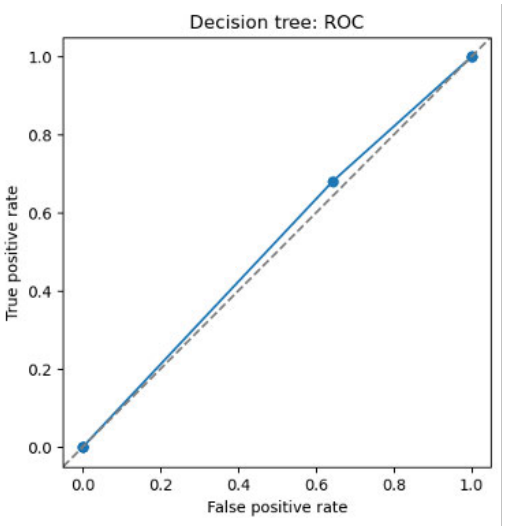

Figure 6: The ROC curve for our best decision tree classifier. We obtained this plot by varying the posterior threshold from 0 to 1 in increments of 0.1.

Table 1: Our final evaluation of our best models from cross validation using the withheld 20% test data. TRAINING/TEST ERR. = training/test error (mean squared error for regression, classification error for classification.

MODEL	PRE-PROCESSING	TRAINING ERR.	TEST ERR.
LASSO regression (L1 penalty = 0.001)	NV2 + C1 + GE2	0.0121	0.0150
LASSO classification (L1 penalty = 1)	NV3 + C1 + GE1	17.62%	20.42%
Decision tree (Minimum impurity decrease = 0.01)	NV2 + C2 + GE2	35.04%	37.92%

4.1 Challenges for classification

Due to the inherent nature of this data (specifically, the limited number of genes in the genome as well as the significant imbalance between the number of essential genes that are within the target class versus those outside it [i.e., affected or not affected by CRISPRoff respectively]), training an effective classifier for this data poses a significant challenge for machine learning algorithms. As indicated by the ROC curves, there is no meaningful difference between our models and models that choose their output by chance. This is very likely due to the data containing so few essential genes which are not affected by CRISPRoff (i.e. out-of-class sample points) relative to those which are, thus learning what an out-of-class sample looks like is difficult. Since there already exists data on the approximately 20,000 genes in the human genome, there are simply no additional genes to probe. However, we were limited to only observing the essential genes due to the CRISPRoff scores only being relevant for those genes. New experiments would have to be designed and done to increase the number of genes that can be meaningfully sampled.[4] More out-of-class data points could be generated by doing more of the same laboratory experiments but with many replicates, eliminating the need to design de novo experiments and ways to measure CRISPRoff efficacy. However, both of these options would be cost and time intensive.

[4]The original experiment was done by observing cell death ratios when CRISPRoff is applied, which is why only the essential genes have meaningful scores. The more advanced experiment we have in mind would involve sequencing the RNA in cells that have had CRISPRoff applied to them, allowing us to look directly at the degree to which CRISPRoff is silencing particular genes irrespective of whether or not it leads to cell death.

4.2 Important features

As the models are interpretable, we can comment on the features that appear to be important across multiple of our models; however, due to the ROC curves, we do not have high confidence that they are truly indicative of underlying biological phenomena. Nonetheless the following features may be worth investigating further:

- Residing on chromosomes 11 or 17
- Typical degree of H3K27ac presence in promoter region
- Typical expression level

4.3 Future work

Collecting more data Most pressingly, we would like to either collect more wet lab data, or find a workable solution with the available data to improve the balance between the two class labels as described in Section 4.1. We would also like to bolster our number of gene features (as described in Section 2.1.2) by aggregating more existing data from the internet.

Comparing to non-interpretable models We would also like to compare our results to non-interpretable models, especially:

- k-nearest neighbors
- Random forest
- Shallow neural network (we will ideally one day have tens of thousands of sample points, but that is likely not enough to train a deep neural network)

Such a comparison would show the limit of how much signal we can extract from the data, and, correspondingly, an upper limit on how predictive we can hope to make our interpretable models.

Improving existing models Finally, we conclude with an idea to improve our existing classifiers, which is to balance (or otherwise compensate for) the number of positive and negative examples using the existing data. A total of approximately 80% of both the training and test data were positive examples, meaning that a classifier that always predicts that CRISPRoff will work well would have only approximately 20% test error, which is comparable to our best classifier (and hence the ROC curve appearing to be a line with slope 1). A potential solution with the currently available data could be to oversample the out-of-class data points to help balance the classes. Another possible solution could be to generate synthetic out-of-class data points based on the limited number of out-of-class points we do have. If we had more experimental data, we could also simply sample the same number of in-class and out-of-class points from the dataset.

Acknowledgments

We thank James Nuñez for helpfully pointing out some of the gene features and corresponding datasets we used.

References

[1] G. J. Knott and J. A. Doudna, "CRISPR-Cas guides the future of genetic engineering," *Science (New York, N.Y.)*, vol. 361, pp. 866–869, Aug. 2018.

[2] M. Jost, D. A. Santos, R. A. Saunders, M. A. Horlbeck, J. S. Hawkins, S. M. Scaria, T. M. Norman, J. A. Hussmann, C. R. Liem, C. A. Gross, and J. S. Weissman, "Titrating gene expression using libraries of systematically attenuated CRISPR guide RNAs," *Nature Biotechnology*, vol. 38, pp. 355–364, Mar. 2020.

[3] J. K. Nuñez, J. Chen, G. C. Pommier, J. Z. Cogan, J. M. Replogle, C. Adriaens, G. N. Ramadoss, Q. Shi, K. L. Hung, A. J. Samelson, A. N. Pogson, J. Y. S. Kim, A. Chung, M. D. Leonetti, H. Y. Chang, M. Kampmann, B. E. Bernstein, V. Hovestadt, L. A. Gilbert, and J. S. Weissman, "Genome-wide programmable transcriptional memory by CRISPR-based epigenome editing," *Cell*, vol. 184, pp. 2503–2519.e17, Apr. 2021.

[4] M. Gardiner-Garden and M. Frommer, "CpG Islands in vertebrate genomes," *Journal of Molecular Biology*, vol. 196, pp. 261–282, July 1987.