Machine Learning Engineer Nanodegree

Capstone Proposal

Michel Moreau July 2nd, 2019

Proposal

CellSignal - Disentangling biological signal from experimental noise in cellular images

Introduction

My final project is to participale in the NeurIPS competition on Kaggle called **CellSignal** - Disentangling biological signal from experimental noise in cellular images. More information about this competition is available here:

- Competition's website https://www.rxrx.ai
- Kaggle competition's link: https://www.kaggle.com/c/recursion-cellular-image-classification/overview

Full disclosure: Some text in this proposal will be taken word for word from the competition's websites.

Domain Background

Recursion Pharmaceuticals, creators of the industry's largest dataset of biological images, generated entirely in-house, believes AI has the potential to dramatically improve and expedite the drug discovery process. More specifically, machine learning could help understand how drugs interact with human cells.

This competition is designed to disentangle experimental noise from real biological signals. The goal is to classify images of cells under one of 1,108 different genetic perturbations, and thus eliminate the noise introduced by technical execution and environmental variation between [drug] experiments.

Datasets and Inputs

The data is available on the Kaggle's competition site https://www.kaggle.com/c/recursion-cellular-image-classification/data .

One of the main challenges for applying AI to biological microscopy data is that even the most careful replicates of a process will not look identical. This dataset challenges you to develop a model for identifying replicates that is robust to experimental noise.

The same siRNAs (effectively genetic perturbations) have been applied repeatedly to multiple cell lines, for a total of 51 experimental batches. Each batch has four plates, each of which has 308 filled wells. For each well, microscope images were taken at two sites and across six imaging channels. Not every batch will necessarily have every well filled or every siRNA present.

For more information about the dataset, see the competition's website or the Kaggle link above.

Solution Statement

The solution for this problem will likely be resolved with the type of model architecture used in computer vision and image classification, e.g convolutional neural networks. This is a multiclass classification problem, but algorithms and model architectures we have seen in the dogs classification project https://github.com/MichelML/udacity-dog-project/, such as VGG-16 and ResNet-50, will be considered for this project.

Benchmark Model

Related to the solution statement, ResNet-50 will likely be our benchmark model, having achieved very good results in multiclass image classification problems in the recent past (see source), having reached 98.87% accuracy when classifying histopathology images. ResNet-50 will be measured with multiclass accuracy, which is the same measure the competition's evaluation uses.

Evaluation Metrics

Submissions will be evaluated on Multiclass Accuracy, which is simply the average number of observations with the correct label.

Submission File

For each id_code in the test set, we will predict the correct siRNA. As per the competition's indications, The file should contain a header and have the following format:

```
id_code,sirna
HEPG2-08_1_B03,911
HEPG2-08_1_B04,911
etc.
```

Project Design

N.B. This design is inspired by Appendix B of the book Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools by Aurélien Géron

1. Frame the problem and look at the big picture.
2. Get the data.
3. Explore the data to gain insights.
4. Prepare the data to better expose the underlying data patterns to Machine Learning algorithms.
5. Explore many different models and short-list the best ones.
6. Fine-tune your models and combine them into a great solution.
7. Present your solution.

Frame the problem and look at the big picture

1. Define the objective in business terms.
2. How will our solution be used?
3. What are the current solutions/workarounds (if any)?

Get the data

1. Get the data from the Kaggle's competition.
2. Convert the data to a format we can easily manipulate (without changing the data itself).

Explore the data

1. Create a copy of the data for exploration (sampling it down to a manageable size if necessary).
2. Create a Jupyter notebook to keep record of our data exploration.
3. Study each attribute and its characteristics:
   - Name
   - Type (categorical, int/float, bounded/unbounded, text, structured, etc.)
   - % of missing values
   - Noisiness and type of noise (stochastic, outliers, rounding errors, etc.)
   - Possibly useful for the task?
   - Type of distribution (Gaussian, uniform, logarithmic, etc.)
4. Visualize the data.
5. Identify the promising transformations we may want to apply.
6. Identify extra data that would be useful (go back to "Get the Data" on page 502).
7. Document what we have learned.

Prepare the data

1. Data cleaning:
   - Fix or remove outliers (optional).
   - Fill in missing values (e.g., with zero, mean, median...) or drop their rows (or columns).
2. Feature selection (optional):
   - Drop the attributes that provide no useful information for the task.
3. Feature engineering, where appropriates:
   - Discretize continuous features.
   - Decompose features (e.g., categorical, date/time, etc.).
   - Add promising transformations of features (e.g., log(x), sqrt(x), x^2, etc.).
   - Aggregate features into promising new features.
4. Feature scaling: standardize or normalize features.

Short-list promising models

1. Train many quick and dirty for each CNN architectures identified, using standard parameters.
2. Measure and compare their performance.
   - For each model, use N-fold cross-validation and compute the mean and standard deviation of their performance.
3. Analyze the most significant variables for each algorithm.
4. Analyze the types of errors the models make.
   - What data would a human have used to avoid these errors?
5. Have a quick round of feature selection and engineering.

6. Have one or two more quick iterations of the five previous steps.
7. Short-list the top three to five most promising models, preferring models that make different types of errors.

Fine-Tune the System

1. Fine-tune the hyperparameters using cross-validation.
    - Treat our data transformation choices as hyperparameters, especially when we are not sure about them (e.g., should I replace missing values with zero or the median value? Or just drop the rows?).
    - Unless there are very few hyperparamter values to explore, prefer random search over grid search. If training is very long, we may prefer a Bayesian optimization approach (e.g., using a Gaussian process priors, as described by Jasper Snoek, Hugo Larochelle, and Ryan Adams (https://goo.gl/PEFfGr))
2. Try Ensemble methods. Combining our best models will often perform better than running them invdividually.
3. Once we are confident about our final model, measure its performance on the test set to estimate the generalization error.

Present our solution

1. Document what we have done.
2. Create a nice presentation.
    - Make sure we highlight the big picture first.
3. Explain why our solution achieves the business objective.
4. Don't forget to present interesting points we noticed along the way.
    - Describe what worked and what did not.
    - List our assumptions and our system's limitations.
5. Ensure our key findings are communicated through beautiful visualizations or easy-to-remember statements (e.g., "the median income is the number-one predictor of housing prices").