The universal workflow of machine learning

Francois Chollet

Defining the problem and assembling a dataset

- First, you must define the problem at hand
  - What will your input data be? What are you trying to predict? You can only learn to predict something if you have available training data
  - What type of problem are you facing? Is it binary classification? Multiclass classification? Single or multiple label? Scalar regression? Identifying the problem type will guide your choice of model architecture, loss function, and so on
- The hypotheses you make at this stage
  - **You hypothesize that your outputs can be predicted given your inputs**
  - **You hypothesize that your available data is sufficiently informative to learn the relationship between inputs and outputs**

The workflow

Choosing a measure of success

- You must define what you mean by success—accuracy? Precision and recall? Customer-retention rate?
- Your metric for success will guide the choice of a loss function: what your model will optimize. It should directly align with your higher-level goals, such as the success of your business
  - For balanced-classification problems, where every class is equally likely, *accuracy* and *area under the receiver operating characteristic curve* (ROC AUC) are common metrics
  - For class-imbalanced problems, you can use precision and recall
  - For ranking problems or multilabel classification, you can use mean average precision
- Data science competitions on Kaggle showcase a wide range of problems and evaluation metrics

Deciding on an evaluation protocol

- Common evaluation protocols
  - Maintaining a hold-out validation set—The way to go when you have plenty of data
  - Doing K-fold cross-validation—The right choice when you have too few samples for hold-out validation to be reliable
  - Doing iterated K-fold validation—For performing highly accurate model evaluation when little data is available

Preparing your data

- Format your data in a way that can be fed into a machine-learning model
    - If different features take values in different ranges (heterogeneous data), then the data should be normalized
    - You may want to do some feature engineering (add features or remove features), especially for small-data problems

Developing a model that does better than a baseline

- Your goal at this stage is to achieve statistical power: that is, to develop a small model that is capable of beating a dumb baseline. In the MNIST digit-classification example, anything that achieves an accuracy greater than 0.1

- It’s not always possible to achieve statistical power. If you can’t beat a random baseline after trying multiple reasonable architectures, it may be that the answer to the question you’re asking isn’t present in the input data

Scaling up: developing a model that overfits

- Once you've obtained a model that has statistical power, the question becomes, is your model sufficiently powerful? Does it have enough parameters to properly model the problem at hand?
- To figure out how big a model you'll need, you must develop a model that overfits. For example, in neural networks:
  - Add layers
  - Make the layers bigger
  - Train for more epochs
- When you see that the model's performance on the validation data begins to degrade, you've achieved overfitting

Regularizing your model and tuning your hyperparameters

- This step will take the most time: you'll repeatedly modify your model, train it, evaluate on your validation data, modify it again, and repeat, until the model is as good as it can get. These are some things you should try:
  - Add regularization
  - Try different architectures (add or remove layers, add or remove polynomial features)
  - Try different hyperparameters (such as the number of units per layer, or the regularization parameter) to find the optimal configuration
  - Optionally, iterate on feature engineering: add new features, or remove features that don't seem to be informative
- Once you've developed a satisfactory model configuration, you can train your final production model on all the available data (training and validation) and evaluate it one last time on the test set

The universal workflow of Machine Learning

1. Choosing a measure of success
2. Deciding on an evaluation protocol
3. Preparing your data
4. Developing a model that does better than a baseline
5. Scaling up: developing a model that overfits
6. Iteration: Regularizing your model and tuning your hyperparameters

References

- Francois Chollet; Deep Learning with Python; Manning Publications, 2017