Guide to Writing a Machine Learning Methods Section

Peter Sadowski

February 20, 2023

The following is simple guide to describing a typical machine learning project. This can be used when writing up a class project or the Methods section of a research paper.

Step-by-Step

1. **Introduce the problem.** What problem are you trying to solve? What are the inputs and outputs of your model?

2. **Introduce the dataset.** Machine learning is about learning from data, and the model will only be as good as the dataset. Clearly explain where your data came from.

3. **Introduce the model.** Which machine learning model(s) are you using? Is there any particular reason?

4. **Specify features and pre-processing.** Describe the feature representation and any transformations applied. This may include:

   - Feature units, e.g. inches, meters, or unit-less.
   - Feature properties, e.g. categorical, ordinal, integer, real-valued, one-hot.
   - Transformations, e.g. min-max scaling, standardization, $\log(x)$ or $\log(x+1)$.
   - Missing data methods, e.g. removal, padding, interpolation, imputation, iterative imputation.
   - Augmentation, e.g. gaussian noise.

5. **Specify data splits and how they are used.**

   Example: *The total dataset contained 1,000,000 examples. The data was randomly permuted, then divided into three subsets: 60% training, 20% validation, and 20% test. Models were trained on the training set, while the validation set was used for early stopping, hyperparameter optimization, and model selection. The test set was used to evaluate the final model.*

6. **Specify the hyperparameter search space.** This is the list of hyperparameters that were optimized, and the range of values that were explored. This can be difficult to describe succinctly since hyperparameter tuning is usually an iterative process involving the experimenter. Ideally, you use a hyperparameter optimization framework like Sherpa [Hertel et al., 2020], but in the case where hyperparameter optimization was mostly done by hand, you can simply state the range of values you explored (min and max) for each hyperparameter and the total number of models you tried.

   Example: *For the K-Nearest Neighbor classifier we tried different values of K and different distance metrics. We tried all odd values of K from the set of integers between 1 and 99, {1,3,5,...,99}. We tried the L1 and L2 distance metrics.*

7. **Explain how hyperparameters were optimized.** Explain if you tuned the hyperparameters by hand, or exhaustively tested every hyperparameter combination in the search space. This can be a simple statement of the metric and validation set used. State any optimization algorithms you used, e.g.

Random Search, Grid Search, Bayesian Optimization, Population-Based Training, etc. (see algorithms in Hertel et al. [2020]).

Examples:

(a) *After trying all combinations of hyperparameters in the search space, the model with the highest accuracy on the validation set was selected.*

(b) *A total of 50 different models were trained, with random combinations of hyperparameters selected from the search space. The model with the highest validation set MSE was selected and evaluated on the test set.*

(c) *Hand-tuning was used to train a total of 20 different models with different hyperparameters. The model with the best validation set AUROC was selected and evaluated on the test set.*

8. **Evaluate model on clean test set.** When quantifying performance, remember to specify the *metric* and *dataset* for every number you present.

   Example:

   (a) *The performance of the final model on the test set was .98 AUROC.*

   (b) The model achieved an accuracy of 80% on the held-out test set.

9. **Explain any differences in the train/test datasets.** If possible, provide justification for why you expect the model to generalize despite differences.

   Example:

   (a) *The test set is from a later time than the training set, so data drift could harm model performance.*

References

L. Hertel, J. Collado, P. Sadowski, J. Ott, and P. Baldi. Sherpa: Robust hyperparameter optimization for machine learning. *SoftwareX*, 12(100591), 2020. doi: 10.1016/j.softx.2020.100591.