Universal workflow of machine learning

Chengheri BAO, December 2019

1. Defining the problem and assembling a dataset
2. Choosing a measure of success
3. Deciding on an evaluation protocol
4. Preparing your data
5. Developing a model that does better than a baseline
6. Scalling up: developing a model that overfits
7. Regularizing your model and tuning your hyperparameters

1. Defining the problem and assembling a dataset

- What will your input data be?
- What are you trying to predict?
- What type of problem are you facing?
  - Is it binary classification? Multiclass classification? Multicalss, multilable classification?
  - Scalare regression? Vector regression?
  - Clustering?
  - Generation?
  - Reinforcement learning?

Hypothesises:
- I hypothesize that my outputs can be predicted given my inputs.
- I hypothesize that my available data is sufficiently informative to learn the relationship between inputs and outputs.

Attention!
- Some problems can not be solved by just having the data. For example, predict the movements of a stock on the stock market given its recent history.
- Nonstationary problems such as clothes buying over the scale of a few months, in this case, the right move is to constantly retrain the model on data from the recent past, or gather data at a timescale where the problem is stationary to capture periodical variation, a few year's worth of data in this example (make sure to make the time of the year an input of the model).

2. Choosing a measure of success

- Accuracy?
- Precision and recall?
- Customer retention?

==> guide the choice of a loss function

Problem type	Metric
balanced classification	accuracy; ROC AUC
imbalanced-class classification	precision, recall
ranking problems	mean average precision
multilabel classification	mean average precision

3. Deciding on an evaluation protocol

Evaluation protocol	Usecase
hold-out validation set	when you have **plenty** of data
K-fold cross validation	when you have **too few samples** for hold-out validation to be reliable
iterated K-fold cross validation	for performing **highly accurate** model evaluation when **little data** is available

4. Preparing your data

I. Format data as tensors
II. Values of these tensors should scalped to small values ([0,1] or [-1,1])
III. Normalize data, if different features take different value ranges (heterogeneous data)
IV. Feature engineering, especially for small-data problems

5. Developing a model that does better than a baseline

The goal is to achieve statistical power.

- If you can't beat a random baseline after trying multiple reasonable architectures, you may need to question your hypothesis.
- If it goes well, then make three key choices to build your first model:
  - Last-layer activation
  - Loss function
  - Optimization configuration

Problem type	Last-layer activation	Loss function	Opimization configuration
Binary classification	sigmoid	binary_crossentropy	rmsprop, default learning rate
Multiclass, singl-label classification	softmax	categorical_crossentropy	rmsprop, default learning rate
Multiclass, multi-label classficiation	sigmoid	binary_crossentropy	rmsprop, default learning rate
Regresstion to arbitrary values	None	mse	rmsprop, default learning rate
Regresstion to values between 0 and 1	sigmoid	mse or binary_crossentropy	rmsprop, default learning rate

6. Scalling up: developing a model that overfits

To find the border between overfitting and underfitting, you need to overfit first:

- Add layers
- Make the layers bigger
- Train for more epochs

Then regularize and tune the model.

Always monitor the training loss and validation loss, training and validation metrics => if performance on validation data degrade, then it overfits

7. Regularizing your model and tuning your hyperparameters

- Add dropout
- Add or remove layers
- Add L1/L2 regularization
- Try different numbers of units per layer

Chengheri BAO LinkedIn chengheri.bao@gmail.com +33 7 60 45 88 32

- Try different learning rates
- Add new features or remove features that are not informative

Attention!
Information can leak into the model from validation process (model overfits on validation data). To check it, train the final model on all the data (training and validation set) and evaluate it one last time on the test set. If the performance on test set is significantly worse than that on validation set ==>

- validation process wasn't reliable
- model is overfitting to the validation data

Solution: iterated K-fold validation

Chengheri BAO LinkedIn chengheri.bao@gmail.com +33 7 60 45 88 32