Deep Learning

AlejandroMllo

This document serves as a very brief summary of the topics covered in each chapter of the book Deep Learning [1].

Disclaimer: This document is completely extracted from [1], the author does not attribute any ownership over the material.

11 Practical Methodology

- Practical design process:
  - Determine your goals.
    - Which error metric to use (usually different from the cost function used to train the model).
    - What is the expected performance.
  - Establish a working end-to-end pipeline as soon as possible, including the estimation of the appropriate performance metrics.
    - Choose the general category of model based on the structure of the data.
    - Optimization algorithm: SGD with momentum with a decaying learning rate; or ADAM.
  - Instrument the system well to determine bottlenecks in performance.
  - Repeatedly make incremental changes based on specific findings from the instrumentation.
- Bayes error defines the minimum error rate that can be achieved.
- Precision: fraction of detections reported by the model that were correct.
- Recall: fraction of true events that were detected by the model.
- To summarize the performance of a classifier, it is possible to convert precision p and recall r into an F-score given by: $F = \frac{2pr}{p+r}$. Another option is to report the total area lying beneath the PR curve.
- In some applications, it is possible for the ML system to refuse to make a decision.
- Coverage: fraction of examples for which the ML system is able to produce a response.
- If performing supervised learning (SL) with fixed-size vectors as input, use a feedforward network with fully connected layers. If the input has known topological structure, use a CNN. In these cases, start by using some kind of piecewise linear unit (ReLUs or their generalizations). If the input or output is a sequence, use a gated recurrent net (LSTM or GRU).
- Batch normalization can have a dramatic effect on optimization performance, especially for CNNs and networks with sigmoidal performance. Sometimes reduces generalization error and allows dropout to be omitted.
- Early stopping should be used almost universally.
- Dropout is an excellent regularizer compatible with many models and training algorithms.
- If the application is in a context where unsupervised learning (UL) is known to be important, then include it in the first end-to-end baseline. Otherwise, only use UL in the first attempt if the task to be solved is unsupervised.
- It is often much better to gather more data than to improve the learning algorithm.
- If performance on the training set is poor, improve the learning algorithm (parameter tunning, increase size of the model, etc). If this does not work, the problem might be the quality of the data.
- If performance on the training set is acceptable, but the test set performance is much worse, then gather more data (an alternative is to reduce the size of the model and/or improve regularization). If it is not possible, improve the learning algorithm itself (but this becomes the domain of researchers).
- Choosing the hyperparameters manually requires understanding what the hyperparameters do and how ML models achieve good generalization. Automatic hyperparameter selection algorithms greatly reduce the need to understand these ideas, but they are often much more computationally costly.

- Effective capacity: the representational capacity of the model, the ability of the learning algorithm to successfully minimize the cost function used to train the model, and the degree to which the cost function and training procedure regularize the model.
- The generalization error typically follows a U-shaped curve when plotter as a function of one of the hyperparameters. Somewhere in the middle lies the optimal model capacity, which achieves the lowest possible generalization error, by adding a medium generalization gap to a medium amount of training error.
- The learning rate is perhaps the most important hyperparameter. It controls the effective capacity of the model, because it is highest when the learning rate is correct. It has a U-shaped curve for training error.
- Tuning the hyperparameters other than the learning rate requires monitoring both training and test error, then adjusting its capacity appropriately.
- Tuning hyperparameters with grid search: for each hyperparameter, the user selects a small finite set of values to explore. The grid search algorithm then trains a model for every joint specification of hyperparameter values in the Cartesian product of the set of values for each individual hyperparameter. The experiment that yields the best validation set error is then chosen as having found the best hyperparameters.
- Another hyperparameter tuning technique is random search, where a PD distribution is defined in a specified search range and according to the characteristics of each hyperparameter.
- Debugging strategies for NN: either we design a case that is so simple that the correct behavior actually can be predicted, or we design a test that exercises one part of the NN implementation in isolation.
- Some debugging tests include: visualize the model in action; visualize the worst mistakes; reason about software using training and test error; fit a tiny dataset; compare back-propagated derivatives to numerical derivatives; monitor histograms of activations and gradient

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.