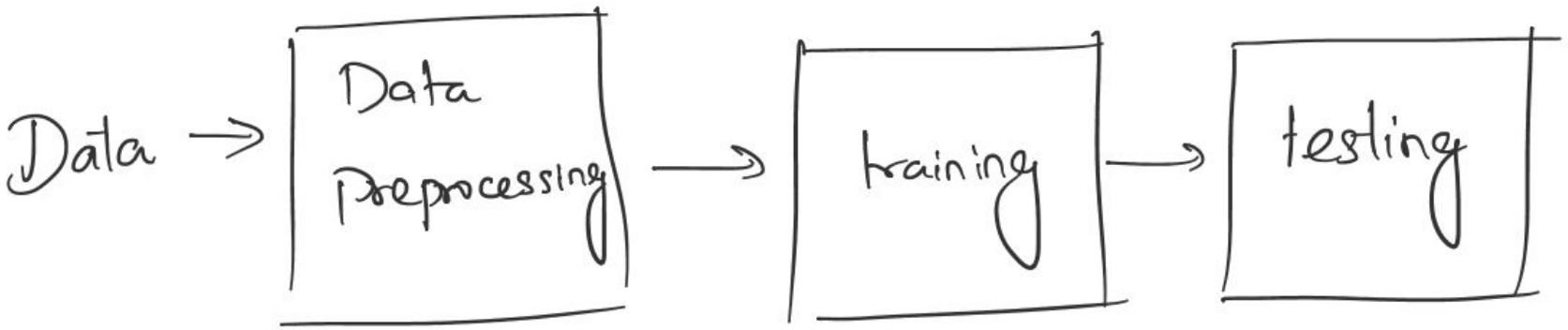
Practical Tips for Machine Learning

Sarath Chandar

Always start with a basic end-to-end pipeline



- Use a simple model and complete the pipeline so that you get some basic results.
- Then you can keep improving the model part of the pipeline with new models.
- You should always have a working system (even if it is a dumb system) from the beginning of the project.

Overfit the model with a single mini-batch

- Take a single mini-batch and see if your model can overfit it.
- If it overfits, then it is a good sign that the implementation is probably correct and the gradient computations are happening correctly.
- If the loss doesn't go to zero:
  - Check for the correctness of the implementation.
  - Check if the gradient computation is implemented correctly by using finite difference approximation.

Hyperparameter tuning

- Always tune your learning rate!
- It is a good practise to list all your hyperparameters in a single config file.
- Spend some time with each hyperparameter by tuning it separately to get a feel of its effect in the learning process.
  - Do this few times and then you will develop an expertise in guessing their effects yourself later.
- Do a proper hyperparameter search.
  - It takes a lot of time and computation!
  - So plan it wisely.

- If there are very few hyperparameters, you can do a grid search.
- If there are too many hyperparameters, use random search.
- If you think that there is a strong structure in the interactions between these hyperparameters, use Bayesian optimization for hyperparameter search.

Organize your experiments

- In a project, you will typically try several changes to the model.
- You will also do a lot of hyperparameter search.
- So it is important to organize your experiments so that you always get a big picture.
- Use experiment manager like Sacred.

Checkpoint your experiments

- What is checkpointing?
  - Checkpointing refers to saving the current state of the experiment which includes
    - State of the model (current parameter, hyperparameter list)
    - State of the optimizer (current book-keeping variables in an optimizer)
    - State of the data-iterator (which minibatch are you currently using)
    - Training statistics (like training loss, validation loss)
    - State of the random number generator.
- Why should you checkpoint?
  - ML experiments take a lot of time. Sometimes your experiments might get killed due to power cuts, unplugging of machines, etc. Checkpointing helps in resuming the experiment from the most recent checkpoint.
  - Checkpointing helps in storing the model and loading it back later for testing purposes.
  - Your jobs needs to be resumable in a cluster or cloud setting since you will have time limit for a single job.

Reproducibility

- Machine learning experiments are often very difficult to reproduce.
- Why do we need reproducibility?
  - To do better science!
  - To verify that others' hypotheses are correct.
  - To build on the top of another person's work.
- Why is it hard?
  - Several sources of randomness.
    - Randomness in the initialization.
    - Randomness in the data iterator.
    - Randomness in sampling processes.
    - Model itself might be probabilistic.
- How to make it easy?
  - Control all the sources of randomness.
  - Always get and set the state of the random components while checkpointing.
  - Tedious process, but once done then all your experiments are reproducible!

How to report your models

- For better reproducibility, you should report your models in a detailed way.
- List all your hyperparameters!
- List the range for each hyperparameter that you used for tuning.
- Always mention the best hyperparameters that you ended up using in your final experiments.
- Write a detailed but precise description of your model.
- You can use the appendix of hyperparameter details and detailed model descriptions.

Training loss and Evaluation metric

- If you are not directly optimizing your evaluation metric, then be careful.
- If your training loss is going down but your validation performance (w.r.t evaluation metric) is very high, then there is a mismatch between the training loss and the evaluation metric.

Overfitting and early stopping

- Always track both training and validation loss.
- When the validation loss starts to increase, the model is starting to overfit.
- You can early stop the training if you see that the validation loss is consistently going up for some 'k' iterations.
- 'k' is your patience number.

First overfit and then regularize

- First try to increase the capacity of the model so that the model can overfit the training data.
- Then add a lot of regularizers to improve the generalization error.

Track the learning

- Always track the train/valid loss.
- Track your gradients to detect abnormalities in the training.
- It is not just enough to print the values in the screen.
- It is easy if you have dynamic visualizations to track how the training progresses.
  - Try Tensorboard or Visdom.

Data

- It is worth the time to do some basic cleaning in your data.
- Always maintain a completely automated preprocessing pipeline.
- It is often useful to normalize the input.

General tips

- Always setup very strong baselines.
- You should first check the performance of a random classifier.
- Then try simple baselines like logistic regression, random forests.
- Do proper hyperparameter tuning for your own baselines.
  - Do justice to them!
- Don't underestimate the training time when you plan your experiments. Always log the training time and profile your code for run time.
- When you report results for a new model, you should
  - Either run multiple runs and report average performance and the error bars.
  - Or runt the same model in multiple datasets.
- If your model has several components, always do some ablation study to understand the effect of each component.

Be careful while using the test set.

- You will often report train/valid/test performance in every epoch.
  - Test performance is reported only for convenience so that you can find the test performance for the selected model without rerunning the model.
  - Always do model selection based on validation set.
- When you design your own ML algorithm or when you are trying to solve a specific problem, you will typically go through the entire ML process multiple times.
- Everytime you repeat the process, you are overfitting to the test set.
- Be careful!

General Tips to become an ML expert

- Read a lot of recent papers.
- Try to implement and reproduce as many papers as you can.
- Participate in a Kaggle competition at least once.

- More about 'what to do next' in the next lecture.