Course 3 - Structuring ML Projects

Mohammad Khalaji

August 22, 2020

1 Week 1

1.1 Orthogonalization

What to tune in order to get specific optimization. For example, in a TV, each knob is dedicated to tuning one and only one parameter. This makes it much easier to tune the TV. As another example, if the steering wheel in a car changed the acceleration too, it would be extremely difficult to control the car properly. In fact, the steering wheel, the break, and the acceleration pedal are orthogonal, and they make it much easier for the driver to control the car.

Chain of assumptions in ML:

- Fit training set well on cost function (in some occasions, comparable to human-level performance)

  How to tune?

  - Adam
  - Bigger network
- Fit dev set well on cost function

  How to tune?

  - Regularization
  - Bigger train set
- Fit test set well on cost function

  How to tune?

  - Bigger dev set
- Perform well in real world

  How to tune?

  - Change dev set
  - Change the cost function

Normally, it is better not to use early stopping because it is a "knob" that modifies the performance on both training set and dev set.

1.2 Single Number Evaluation Metrics

There often is a trade-off between precision and recall, and we care about both of them. As a result, rather than using 2 numbers to compare different models, we only use one number.

$$F1 = \frac{2}{\frac{1}{P} + \frac{1}{R}} \ (Harmonic\ Mean)$$

1.3 Satisficing and Optimizing Metrics

It's not always easy to have one single evaluation metric, while many important metrics are present.

Classifier	Accuracy	Running Time
A	90%	80ms
B	92%	95ms
C	95%	1500ms

It is not a good idea to set $metric = accuracy - 0.5 \times runningTime$. However we could take one approach: maximize the accuracy, and keep the running time below $100ms$. This way, accuracy is a **Optimizing Metric**, whereas running time is a **Satisficing Metric**.

In another example, consider a situation where you are training a wakeword/trigger detector neural network. Although accuracy is very important in this network, the number of false positives is a very important metric too because you don't want too wake your device up without a valid trigger. As a result, your metrics might be like this:

- Maximize accuracy (Optimizing Metric)
- Keep the number of false positives at most 1 in every 24 hours (Satisficing Metric)

1.4 Train/Dev/Test Sets Distribution

Dev and test sets must not come from different distributions. Having dev and test sets from different distributions is like setting a target, spend many month on reaching it, only to realize that you have moved the target to a different position for the testing phase!

1.5 Size of Dev and Test Sets

In the modern era of deep learning, dataset sizes are growing larger and larger. In past, researchers would use a 60%/20%/20% split for Train/Test/Dev sets. However, when datasets have 1000000 examples, for instance, using only 2% of the data for dev and test sets is reasonable because 1 percent of 1000000 still contains a lot of examples.

Size of your test set must be big enough to give us enough confidence about the model.

Sometimes, a Train/Dev split is enough. However, it is not recommended.

1.6 When to Change Dev/Test Sets

For example, when your model performs poorly on specific kinds of data, you must add those kinds to your Dev/Test sets in order to train a more sensitive method. You can also change your metric in order to incorporate your sensitivity to that kinds of data.

1.7 Why Human Level Performance?

Neural net models surpass human level performance quickly, but they struggle while trying to perform even better. This is because of **Bayes Optimal Error**, which is the best that we can possibly get. Since humans are pretty good at doing certain tasks, the gap between human performance and Bayes error is small. As a result, it is difficult for AI models to get closer to Bayes error.

So long as ML is worse than humans, we can do the following:

- Get labeled data from humans
- Analyze bias/variance
- Conduct manual error analysis

1.8 Avoidable Bias

We don't want to do *too well* on the train set. When human error is 1%, a model with 8% error can be still improved. However, when human error on your dataset is 7.5%, a model with 8% might not be bad at all!

Take human error as a proxy for Bayes error. Then decide on whether to improve the bias or the variance.

- If there is a noticeable difference between train error and Bayes error ($\approx$ human error), focus on your bias problem in order to get closer to Bayes error.
- Otherwise, focus on variance problem (if any)

The difference between Bayes error and human error is called the **Avoidable Bias**. It means that there is some bias that you cannot possibly get below, unless you are overfitting, or you're dealing with special cases that are explained later.

1.9 Surpassing Human Level Performance

- Online Advertising
- Product Recommendations
- Logistics (Predicting Transit Time)
- Loan Approvals

Generally, they come from structured data, and they are not *natural perception* problems (unlike image classification or NLP)

1.10 Improving Model Performance

To solve the problem of avoidable bias and getting closer to human-level performance:

- Train a bigger model
- Train longer/better optimization algorithms
- NN architecture/hyperparameters search

To solve the problem of variance:

- More data
- Regularization
- NN architecture/hyperparameters search

2 Week 2