[Summary]ML System Design

Prioritizing What to Work On

Different ways we can approach a machine learning problem:

- Collect lots of data (for example "honeypot" project but doesn't always work)
- Develop sophisticated features (for example: using email header data in spam emails)
- Develop algorithms to process your input in different ways (recognizing misspellings in spam).

It is difficult to tell which of the options will be helpful.

Error Analysis

The recommended approach to solving machine learning problems is:

1. Start with a simple algorithm, implement it quickly, and test it early.
2. Plot learning curves to decide if more data, more features, etc. will help
3. Error analysis: manually examine the errors on examples in the cross validation set and try to spot a trend.

It's important to get error results as a single, numerical value. Otherwise it is difficult to assess your algorithm's performance.

You may need to process your input before it is useful. For example, if your input is a set of words, you may want to treat the same word with different forms

(fail/failing/failed) as one word, so must use "stemming software" to recognize them all as one.

⚠️ Error Metrics for Skewed Classes

It is sometimes difficult to tell whether a reduction in error is actually an improvement of the algorithm.

- For example: In predicting a cancer diagnoses where 0.5% of the examples have cancer, we find our learning algorithm has a 1% error. However, if we were to simply classify every single example as a 0, then our error would reduce to 0.5% even though we did not improve the algorithm.

This usually happens with **skewed classes**; that is, when our class is very rare in the entire data set.

Or to say it another way, when we have lot more examples from one class than from the other class.

For this we can use **Precision/Recall**.

- Predicted: 1, Actual: 1 ––– True positive
- Predicted: 0, Actual: 0 ––– True negative
- Predicted: 0, Actual, 1 ––– False negative
- Predicted: 1, Actual: 0 ––– False positive

Precision: of all patients we predicted where y=1, what fraction actually has cancer?

$$\frac{\text{True Positives}}{\text{Total number of predicted positives}} = \frac{\text{True Positives}}{\text{True Positives} + \text{False positives}}$$

Recall: Of all the patients that actually have cancer, what fraction did we correctly detect as having cancer?

$$\frac{\text{True Positives}}{\text{Total number of actual positives}} = \frac{\text{True Positives}}{\text{True Positives} + \text{False negatives}}$$

These two metrics give us a better sense of how our classifier is doing. We want both precision and recall to be high.

F1 Score: Trading Off Precision and Recall

We might want a **confident** prediction of two classes using logistic regression. One way is to increase our threshold:

$$Predict\ 1\ if: \mathrm{h}_\theta(x) \geq 0.7$$
$$Predict\ 0\ if: \mathrm{h}_\theta(x) < 0.7$$

This way, we only predict cancer if the patient has a 70% chance.

Doing this, we will have **higher precision** but **lower recall** (refer to the definitions in the previous section).

In the opposite example, we can lower our threshold:

$$Predict\ 1\ if: \mathrm{h}_\theta(x) \geq 0.3$$
$$Predict\ 0\ if: \mathrm{h}_\theta(x) < 0.3$$

That way, we get a very **safe** prediction. This will cause **higher recall** but **lower precision**.

The greater the threshold, the greater the precision and the lower the recall.

The lower the threshold, the greater the recall and the lower the precision.

In order to turn these two metrics into one single number, we can compute the **F Score** (or F1 score).

$$F\ Score = 2\frac{PR}{P+R}$$

In order for the F Score to be large, both precision and recall must be large.

We want to train precision and recall on the **cross validation set** so as not to bias our test set.