ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI FORLÌ

91258 / B0385
Natural Language Processing

Lesson 9. Training and Evaluation in Machine Learning

Alberto Barrón-Cedeño
a.barron@unibo.it

30/10/2024

Table of Contents

In part, derived from Appendix D of Lane et al. (2019)

Current Training and Evaluation Cycle

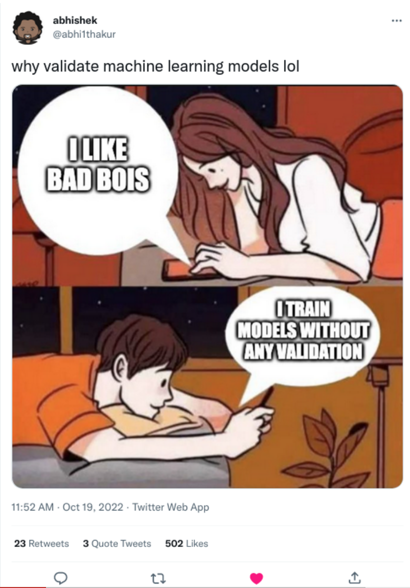
Current Training and Evaluation Cycle

This is what we have been doing so far

1. Train a model m on a dataset C
2. Apply the resulting model m to the same dataset C
3. Compute error or accuracy

This is wrong!

Current Training and Evaluation Cycle



https://twitter.com/abhi1thakur/status/1582870921110016001

Generalisation

A model can generalise if it is able to correctly label an example that is outside of the training set (Lane et al., 2019, 447)

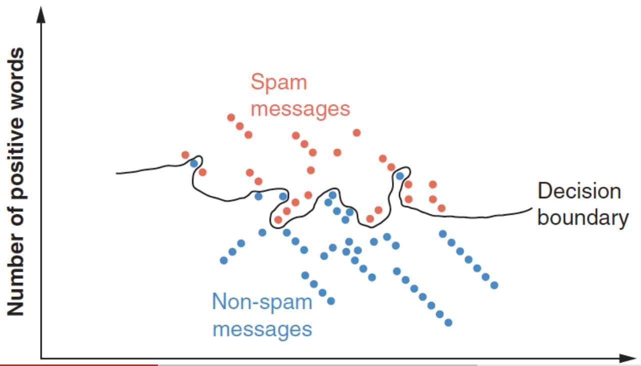
There are two big enemies of generalisation:

- Overfitting
- Underfitting

Overfitting

A model that predicts perfectly the training examples

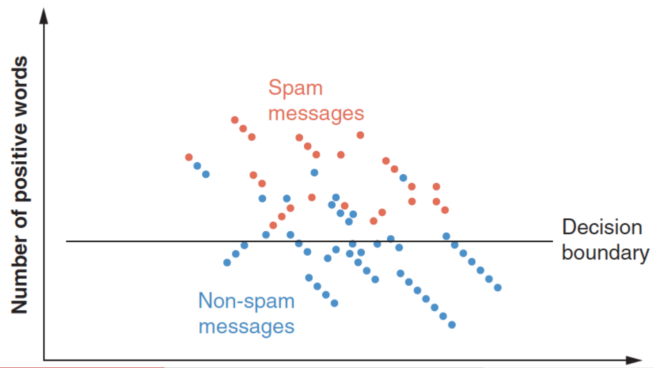
- It lacks capacity to discriminate new data
- In general, it should not be trusted
  Either the problem is trivial or the model/representations do no generalise)



Underfitting

A model that makes many mistakes, even on the training examples

- It lacks capacity to discriminate new data (as well!)
- In general, it should not be trusted
  Either the problem is too difficult or the model/representations are not enough



Fitting (Generalising)

A model that, even if it makes some mistakes on the training examples, makes about the same amount of mistakes on the testing examples

- It has the capacity to discriminate (generalise on) new data
- In general, it could be trusted
  The problem is reasonable and the model/representations are good enough

Data Partitioning

Data Partitioning

So far, we have used all the data available for both training and testing

This is wrong!

Instead, we need to partition it by...

- Held out
- Cross-fit

Always shuffle the data first

Data Partitioning: held out

Fixing three data partitions: one specific purpose each

Training	Instances used to train the model
Development	Instances to optimise the model
Test	Instances to test the model

1: **while** performance on dev $<$ reasonable **do**
2: adjust configuration
3: train m on the training partition
4: evaluate the performance of m on the dev partition
5: re-train m on train+dev partition ▷ only once
6: evaluate the performance of m on the test partition ▷ only once

Data Partitioning: held out

Adjust configuration

- Adapt representation
- Change learning parameters
- Change learning model

Reasonable performance

- A pre-defined value is achieved (e.g., better than a reasonable baseline)
- The model has stopped improving (convergence)

Evaluate on Test

- Carried out only once, with the best model on development
- Keep the test aside (and don't look at it) during tuning

Data Partitioning: held out

Typical distribution

Mid-size data

training	70%
development	15%
testing	15%

Large data

training	90%
development	5%
testing	5%

Often, the partitions have been predefined by the people behind the data release. In general, if that is the case, stick to that partition

Data Partitioning: k-fold cross validation

Splitting into k folds which play different roles in different iterations

Fold 0 First $|C|/k$ instances
Fold 1 Next $|C|/k$ instances
...
Fold k Last $|C|/k$ instances

```
split C into k partitions
performance = {}
for i in [0, 1, ..., k] do
    training set ← all partitions, except for i
    validation set ← partition i
    train on the training set                    ▷ same as before
    perf = evaluate on the validation set
    performance[i] = perf
overall_performace = avg(performance)
```


Data Partitioning: k-fold cross validation

From https://en.wikipedia.org/wiki/Cross-validation_(statistics)

Data Partitioning: *k*-fold cross validation

Typical evaluation strategies

- Compute mean and standard deviation over the k experiments (sd is important: if it is too high, the model is to volatile, or the partitions are not representative)
- Train a *new* model on all folds, with the best configuration, and test on an extra test set

Data Partitioning: leave-one-out cross validation

An extreme case in which $k = |C|$

- Reasonable when the data is relatively small
- It might be too expensive

Imbalanced Data

Imbalanced Data: example

Imagine you want to train a model that differentiates dogs and cats (Lane et al., 2019, pp. 452–453)

dogs	200 pictures
cats	20,000 pictures

```
def dogs_vs_cats(x):
  return "cat"
```

- A model predicting always "cat" will be correct 99% of the time
- Such model wont be able to predict any "dog"
- Such model is useless

Can you think of this kind of data/problem in real life?

Dealing with Imbalanced Data

Oversampling
Repeating examples from the under-represented class(es)

Undersampling
Dropping examples from the over-represented class(es)

Data Augmentation[1]
Produce new instances by perturbation of the existing ones or from scratch

Distant Supervision[2]
Use some labeled training data (on a related task) to label unlabelled data, producing new (noisy) entries

[1]For instance, by means of round-trip translation (Tedesco, 2022) or by active learning (Zhang, 2021)
[2]As in *proppy* for propaganda identification (Barrón-Cedeño et al., 2019)

Performance Metrics

Performance Metrics
True, false, positive, and negative

Confusion matrices

		predicted label positive	predicted label negative
true label	positive	true positive	false positive
	negative	false negative	true negative

Performance Metrics
Accuracy

		predicted label positive	predicted label negative
true label	positive	true positive	false positive
	negative	false negative	true negative

$$Acc = \frac{|\text{true positives}| + |\text{true negatives}|}{|\text{all instances}|} \tag{1}$$

Performance Metrics
Precision

		predicted label	
		positive	negative
true	positive	true positive	false positive
label	negative	false negative	true negative

$$P = \frac{|\text{true positives}|}{|\text{true positives}| + |\text{false positives}|} \tag{2}$$

Performance Metrics
Recall

		predicted label	
		positive	negative
true	positive	true positive	false positive
label	negative	false negative	true negative

$$R = \frac{|\text{true positives}|}{|\text{true positives}| + |\text{false negatives}|} \tag{3}$$

Performance Metrics
F_1-measure

		predicted label	
		positive	negative
true	positive	true positive	false positive
label	negative	false negative	true negative

Combining Eqs. (2) and (3):

$$F_1 = 2\frac{P \cdot R}{P + R} \tag{4}$$

Let us see

Performance Metrics
More on Evaluation

- If the problem is multi-class, the performance is computed on all the classes and (often) combined
  - Micro-averaged
  - Macro-averaged
- If the problem is sequence tagging (e.g., named-entity recognition), the items are characters or words, not documents
- If the problem is not classification, but regression, we need root mean square error (or mean absolute error)
- If the problem is ∼text generation (e.g., machine translation), we need other evaluation schema

References

Barrón-Cedeño, A., I. Jaradat, G. Da San Martino, and P. Nakov
2019. Proppy: Organizing the news based on their propagandistic content. *Information Processing & Management*, 56(5):1849–1864.

Lane, H., C. Howard, and H. Hapkem
2019. *Natural Language Processing in Action*. Shelter Island, NY: Manning Publication Co.

Tedesco, N.
2022. *Round-Trip Translation: A Method for Estimating Revision and Editing Difficulty of English as a Lingua Franca Academic Texts*. Master spectra, Department of Interpreting and Translation, Università di Bologna, Forlì, Italy.

Zhang, S.
2021. *Emotion Identification in Italian Opera*. Master spectra, Department of Interpreting and Translation, Università di Bologna, Forlì, Italy.