

92586 Computational Linguistics

Lesson 6. Training and Evaluation in Machine Learning

Alberto Barrón-Cedeño

Alma Mater Studiorum-Università di Bologna

a.barron@unibo.it

@albarron_

19/03/2020



Mid-Term Pause

- Training and Testing ML Models

Table of Contents

- 1 Current Training and Evaluation Cycle
- 2 Data Partitioning
- 3 Imbalanced Data
- 4 Performance Metrics

Part of 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!

Generalisation

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

Generalisation

A model generalize 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

Overfitting

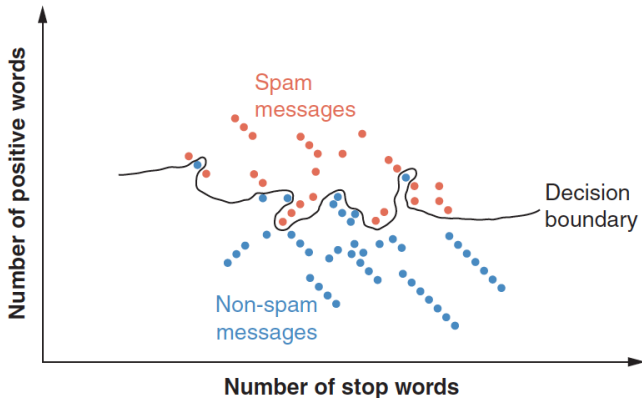
A model that predicts perfectly the training examples

- It lacks capacity to discriminate new data
- In general you should not trust it (either your problem is trivial or your model/representations do not generalise)

Overfitting

A model that predicts perfectly the training examples

- It lacks capacity to discriminate new data
- In general you should not trust it (either your problem is trivial or your model/representations do not generalise)



Underfitting

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

Underfitting

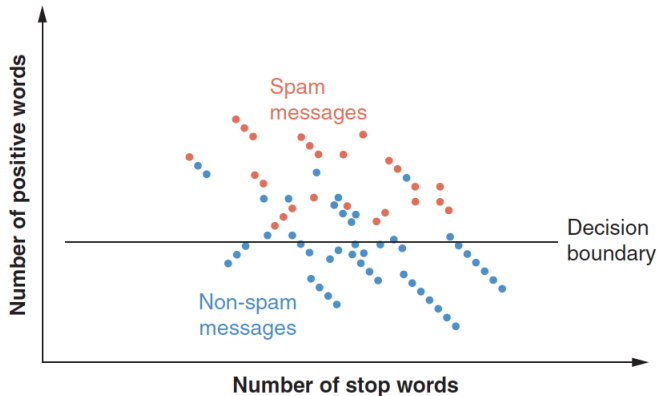
A model that makes many mistakes, even on the training examples

- It lacks capacity to discriminate new data (as well!)
- In general you should not trust it (your problem is too difficult or your model/representations are not enough)

Underfitting

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

- It lacks capacity to discriminate new data (as well!)
- In general you should not trust it (your problem is too difficult or your model/representations are not enough)



Data Partitioning

Data Partitioning

So far, we have used all the data for training

Data Partitioning

So far, we have used all the data for training

Instead, we need to partition it by...

- Held out
- Cross-fit

Data Partitioning

So far, we have used all the data for training

Instead, we need to partition it by...

- Held out
- Cross-fit

Always shuffle the data first

Data Partitioning: held out

Fixing three data partitions for specific purposes

Training Instances used to train the model

Development Instances to optimise the model

Test Instances to test the model

Data Partitioning: held out

Fixing three data partitions for specific purposes

Training Instances used to train the model

Development Instances to optimise the model

Test Instances to test the model

Until performance on dev is ‘‘reasonable’’:

adjust configuration

train the model on the training partition

apply the model to the dev partition

evaluate the performance on the dev partition

Evaluate the model on the test partition

Data Partitioning: held out

Fixing three data partitions for specific purposes

Training Instances used to train the model

Development Instances to optimise the model

Test Instances to test the model

```
Until performance on dev is ‘‘reasonable’’:  
    adjust configuration  
    train the model on the training partition  
    apply the model to the dev partition  
    evaluate the performance on the dev partition
```

```
Evaluate the model on the test partition
```

Data Partitioning: held out

Adjust configuration

- Adapt representation
- Change learning parameters
- Change learning model

Data Partitioning: held out

Adjust configuration

- Adapt representation
- Change learning parameters
- Change learning model

Reasonable performance

- A pre-defined value is achieved
- The models has stopped improving

Data Partitioning: held out

Adjust configuration

- Adapt representation
- Change learning parameters
- Change learning model

Reasonable performance

- A pre-defined value is achieved
- The models has stopped improving

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%

Data Partitioning: held out

Typical distribution

Mid-size data

training 70%

development 15%

testing 15%

Large data

training 90%

development 5%

testing 5%

Data Partitioning: k -fold cross validation

Splitting into k folds which play different roles in different iterations

Fold 1 First $|C|/k$ instances

Fold 2 Next $|C|/k$ instances

...

Fold k Last $|C|/k$ instances

Data Partitioning: k -fold cross validation

Splitting into k folds which play different roles in different iterations

Fold 1 First $|C|/k$ instances

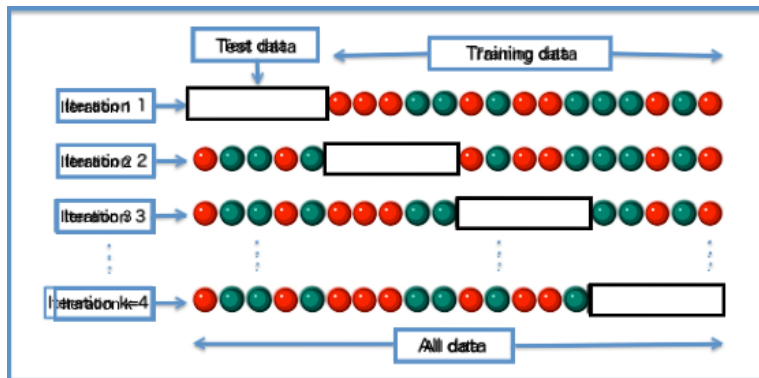
Fold 2 Next $|C|/k$ instances

...

Fold k Last $|C|/k$ instances

```
Split data into k partitions
for i in 1, 2, ..., k:
    training set = all partitions, except i
    validation set = partition i
    run the training/validation process
```

Data Partitioning: k -fold cross validation



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

Data Partitioning: k -fold cross validation

Typical evaluating strategies

- Compute mean and standard deviation over the k experiments
- Train a new model on all folds, with the best configuration, and test on an extra test set

Data Partitioning: k -fold cross validation

Typical evaluating strategies

- Compute mean and standard deviation over the k experiments
- 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 $k = |C|$

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

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

- A model predicting **always** “cat” will be correct 99% of the times
- Such model won't be able to predict any “dog”

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

- A model predicting **always** “cat” will be correct 99% of the times
- Such model won't be able to predict any “dog”

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

Dealing with Imbalanced Data

Oversampling

Repeating examples from the under-represented class(es)

Dealing with Imbalanced Data

Oversampling

Repeating examples from the under-represented class(es)

Undersampling

Dropping examples from the over-represented class(es)

Dealing with Imbalanced Data

Oversampling

Repeating examples from the under-represented class(es)

Undersampling

Dropping examples from the over-represented class(es)

Data Augmentation

Produce new instances by perturbation of the existing ones or from scratch

Dealing with Imbalanced Data

Oversampling

Repeating examples from the under-represented class(es)

Undersampling

Dropping examples from the over-represented class(es)

Data Augmentation

Produce new instances by perturbation of the existing ones or from scratch

Distant Supervision

Use some labeled training data to label unlabelled data, producing new (noisy) entries

Performance Metrics

Performance Metrics

True, false, positive, and negative

		true condition	
		positive	negative
predicted condition	positive	true positive	false positive
	negative	false negative	true positive

Performance Metrics

Accuracy

		true condition	
		positive	negative
predicted condition	positive	true positive	false positive
	negative	false negative	true negative

$$Acc = \frac{|true\ positives| + |true\ negatives|}{|all\ instances|} \quad (1)$$

Performance Metrics

Precision

		true condition	
		positive	negative
predicted condition	positive	true positive	false positive
	negative	false negative	true positive

$$P = \frac{|\text{true positives}|}{|\text{true positives}| + |\text{false positives}|} \quad (2)$$

Performance Metrics

Recall

		true condition	
		positive	negative
predicted condition	positive	true positive	false positive
	negative	false negative	true positive

$$R = \frac{|\text{true positives}|}{|\text{true positives}| + |\text{false negatives}|} \quad (3)$$

Performance Metrics

F_1 -measure

		true condition	
		positive	negative
predicted condition	positive	true positive	false positive
	negative	false negative	true positive

Combining Eqs. (2) and (3):

$$F_1 = 2 \frac{P \cdot R}{P + R} \quad (4)$$


Performance Metrics

F_1 -measure

		true condition	
		positive	negative
predicted condition	positive	true positive	false positive
	negative	false negative	true positive

Combining Eqs. (2) and (3):

$$F_1 = 2 \frac{P \cdot R}{P + R} \quad (4)$$

 Let us see

Performance Metrics

More on Evaluation

- If the problem is multi-class, the performance is computed on all the classes and combined
 - ▶ Micro-averaged
 - ▶ Macro-averaged

Performance Metrics

More on Evaluation

- If the problem is multi-class, the performance is computed on all the classes and combined
 - ▶ Micro-averaged
 - ▶ Macro-averaged
- If the problem is sequence tagging (e.g., plagiarism detection), the items are characters or words, not documents

Performance Metrics

More on Evaluation

- If the problem is multi-class, the performance is computed on all the classes and combined
 - ▶ Micro-averaged
 - ▶ Macro-averaged
- If the problem is sequence tagging (e.g., plagiarism detection), the items are characters or words, not documents
- If the problem is not classification, but regression, we need **root mean square error**

Performance Metrics

More on Evaluation

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

Coming Next

- Back to LSA

References

Lane, H., C. Howard, and H. Hapkem

2019. *Natural Language Processing in Action*. Shelter Island, NY:
Manning Publication Co.