Text Classification with Born's Rule

Linköping University, March 22, 2023

Emanuele Guidotti [1] Alfio Ferrara [2]

[1]University of Neuchâtel, Switzerland

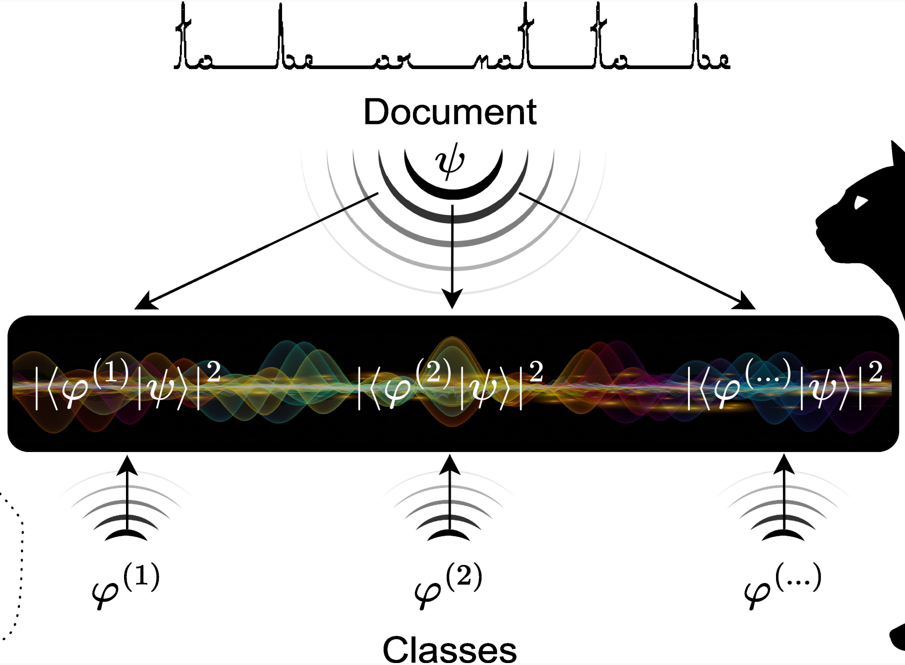
[2]University of Milan, Italy

Table of contents

Text classification

Supervised learning:

1. **Training**. Given documents and classes $\rightarrow$ train the model
2. **Inference**. Given new documents $\rightarrow$ predict their class


to be or not to be
Document
ψ
$|\langle\varphi^{(1)}|\psi\rangle|^2$
$|\langle\varphi^{(2)}|\psi\rangle|^2$
$|\langle\varphi^{(\dots)}|\psi\rangle|^2$
$\varphi^{(1)}$
$\varphi^{(2)}$
$\varphi^{(\dots)}$
Classes

Classification algorithm

Classification algorithm ($x \geq 0$)

Let the feature vector $\mathbf{x}$ contain only non-negative elements, then:

1. Wave function of the document: [1]

$$|\psi\rangle = \sum_j \psi_j |j\rangle = \sum_j \sqrt{x_j} |j\rangle \,. \tag{1}$$

2. Wave function of the k-th class: [2]

$$|\varphi^{(k)}\rangle = \sum_j \varphi_j^{(k)} |j\rangle = \sum_j \sqrt{P_{j|k}} |j\rangle \,. \tag{2}$$

3. Transition probability of the document to the k-th class: [3]

$$u_k = P(\psi \to \varphi^{(k)}) = |\langle \varphi^{(k)} | \psi \rangle|^2 = \left| \sum_j \bar{\varphi}_j^{(k)} \psi_j \right|^2 = \left(\sum_j \sqrt{P_{j|k} x_j} \right)^2 , \tag{3}$$

   and the normalized classification probabilities are $y_k = u_k / \sum_k u_k$.

[1] Obtained by setting the probability of the document to collapse in the j-th word equal to x_j.
[2] Obtained by setting the transition probability from $|\varphi^{(k)}\rangle$ to $|j\rangle$ equal to the conditional probability $P_{j|k}$.
[3] Obtained by applying Born's rule with (1) and (2).

Training

To obtain the conditional probability $P_{j|k}$ in (3) we proceed as follows. Given a training set $\{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1,\ldots,N}$, we normalize each feature vector $\mathbf{x}^{(n)}$ such that it sums up to 1:

$$z_j^{(n)} = \frac{x_j^{(n)}}{\sum_{j'} x_{j'}^{(n)}}. \tag{4}$$

Then, we compute the conditional probability $P_{j|k}$ from the (unnormalized) joint probability P_{jk}:

$$P_{jk} = \sum_n z_j^{(n)} y_k^{(n)}, \quad P_{j|k} = \frac{P_{jk}}{\sum_{j'} P_{j'k}}. \tag{5}$$

Regularization

To regularize the predictions, we re-weight the summation in (3):

$$u_k = \Big(\sum_j H_j \sqrt{P_{j|k} x_j}\Big)^2 , \tag{6}$$

where H_j are entropic weights that range between 0 and 1:

$$H_j = 1 - \frac{\mathcal{H}_j}{\mathcal{H}_{max}} . \tag{7}$$

To simplify ablation studies, we generalize (6) as follows:

$$u_k = \left(\sum_j H_j^h W_{jk}^a x_j^a\right)^{\frac{1}{a}} \quad \text{with} \quad W_{jk} = \frac{P_{jk}}{(\sum_{j'} P_{j'k})^b (\sum_{k'} P_{jk'})^{1-b}}\,, \tag{8}$$

where $a > 0$, $b \geq 0$, and $h \geq 0$ are the model hyper-parameters. The choice $a = \frac{1}{2}$, $b = 1$, and $h = 1$, corresponds to the original model in (6).

- **Local explanation.** The contribution of the j-th feature to the total probability u_k is given by the addend $H_j^h W_{jk}^a x_j^a$ in (8).
- **Global explanation.** The explanation at the class level is obtained by investigating the product $H_j^h W_{jk}^a$ in (8), regardless of the vector $\mathbf{x}$.

Computational complexity

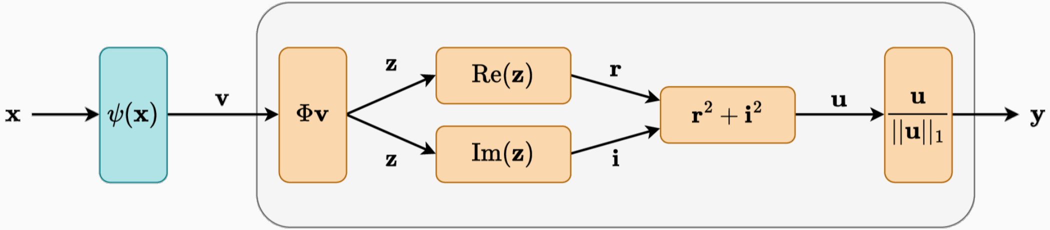
- **Training** $\mathcal{O}(NJK)$. The training time is at most linear in the number of samples (N), in the number of features (J), and in the number of classes (K).
- **Inference** $\mathcal{O}(JK)$. The prediction time is at most linear in the number of features (J), and in the number of classes (K), and it does not depend on the number of training samples (N).

Neural architecture

Neural architecture ($x \in \mathbb{C}$)

1. The wave function of the document is represented with a neural network $\psi_s = \psi_s(\mathbf{x})$ that maps the feature vector $\mathbf{x} \in \mathbb{C}^J$ to the vector of wave coefficients $\psi \in \mathbb{C}^S$.
2. The wave functions of the k-th class is represented with the wave function $|\varphi^{(k)}\rangle = \sum_s \varphi_s^{(k)} |s\rangle$ where the coefficients $\varphi_s^{(k)}$ depend on k and s, but not on $\mathbf{x}$.
3. We use Born's rule to compute the probability of $|\psi\rangle$ to collapse in $|\varphi^{(k)}\rangle$:

$$u_k = P(\psi \to \varphi^{(k)}) = |\langle \varphi^{(k)} | \psi \rangle|^2 = \left| \sum_s \bar{\varphi}_s^{(k)} \psi_s(\mathbf{x}) \right|^2 . \tag{9}$$



Initialization

We initialize the weights Φ_{ks} such that:

1. The wave function $|\varphi^{(k)}\rangle$ has an equal probability to collapse in any state $|s\rangle$.
2. The weights Φ_{ks} are uniformly distributed in the complex circle (isotropy).

$$\Phi_{ks} = \frac{e^{i\theta_{ks}}}{\sqrt{S}} \quad \text{with} \quad \theta_{ks} \sim \mathcal{U}(0, 2\pi). \tag{10}$$

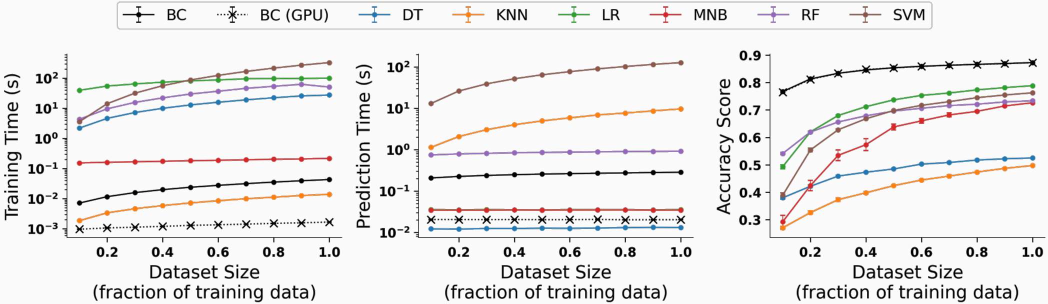
Results

Empirical setup

- **Datasets**: 20Newsgroup (20NG) and Reuters (R8 & R52).
- **Pre-processing**: tokenization using the function nltk.word_tokenize and vectorization with TfidfVectorizer.
- **Algorithms**: Born Classifier (BC), Born Layer (BL), Born Layer initialized with the weights computed by Born Classifier (BC+BL).
- **Baseline**: Decision Tree (DT), K-Nearest Neighbors (KNN), Random Forest (RF), Support Vector Machine (SVM), Multinomial Naive Bayes (MNB), Logistic Regression (LR). For all the algorithms in the baseline, we use the corresponding implementation in scikit-learn.

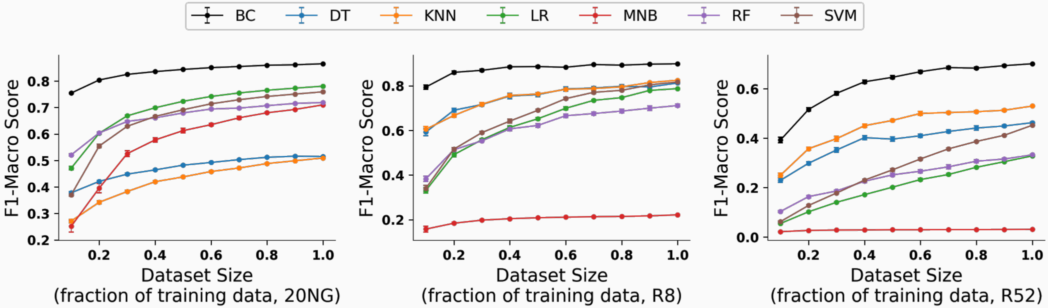
Dataset	Classes	Vocabulary	Train samples	Test samples
20NG	20	204’817	11’314	7’532
R8	8	33’593	5’485	2’189
R52	52	38’132	6’532	2’568

Computational times and accuracy scores on 20Newsgroup



From left to right: training time, prediction time, and accuracy score on the 20Newsgroup dataset (y-axis) for several classifiers, in function of the fraction of data used for training (x-axis). All the classifiers are executed on CPU with default parameters.

F1-macro scores on 20Newsgroup (20NG) and Reuters (R8 & R52)

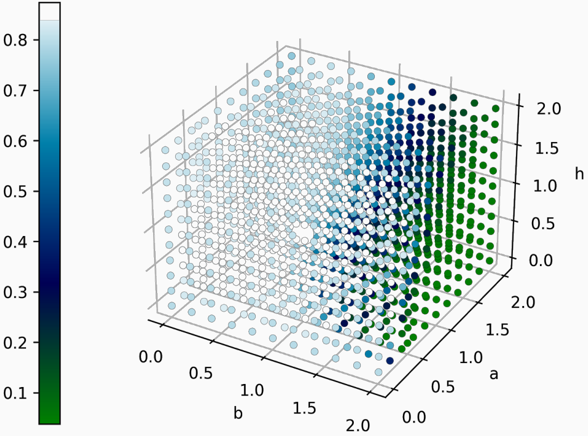


F1-macro score (y-axis) on 20NG, R8, R52, for several classifiers, in function of the fraction of data used for training (x-axis). All the classifiers are executed with default parameters.

Fine-tuned baseline

Accuracy score and runtime for several classifiers on the 20Newsgroup dataset. The runtime is the (CPU) time to optimize the model's hyper-parameters by 5-fold cross-validated grid-search on the training set, plus the time to refit the selected model. The accuracy score is the accuracy achieved by the model on the test set. BC has no hyper-parameters to tune.

Model	Accuracy (%)	Runtime (s)
DT	53.9	5,583.156
KNN	55.6	640.574
RF	77.5	49,686.936
SVM	79.4	45,639.071
LR	82.9	6,066.966
MNB	84.1	15.320
BC	**87.3**	**0.043**



The figure displays the test accuracy scores of the model in (8) for several values of the hyper-parameters a, b, and h. The biggest point identifies the configuration of hyper-parameters $a = \frac{1}{2}$, $b = 1$, $h = 1$, which corresponds to the BC model in (6).

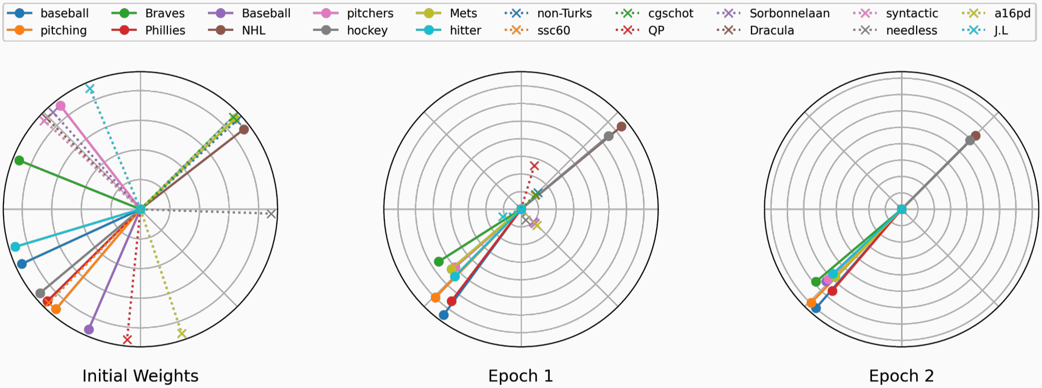
Comparison with literature

	Accuracy (%)			GPU Runtime (s)		
	20NG	R8	R52	20NG	R8	R52
CoNN [1]	83.7	N/A	N/A	120.000	N/A	N/A
TextEnt [3]	84.5	96.7	N/A	923.089	556.020	N/A
TextGCN [4]	86.3	97.1	93.6	1206.372	109.184	186.531
NABoE [2]	86.8	97.1	N/A	152.154	24.110	N/A
DEns [5]	87.1	**97.7**	94.3	N/A	N/A	N/A
BC	87.3	95.4	88.0	**0.001**	**0.001**	**0.001**
BL (1 epoch)	84.6	96.5	87.9	0.347	0.276	0.274
BL (10 epochs)	86.2	96.8	92.6	3.451	2.747	2.723
BL (100 epochs)	87.1	97.1	92.7	34.461	27.452	27.171
BC+BL (1 epoch)	86.9	97.5	91.8	0.348	0.278	0.276
BC+BL (10 epochs)	**87.4**	**97.7**	**95.2**	3.458	2.764	2.724
BC+BL (100 epochs)	**87.4**	97.2	94.4	34.521	27.494	27.124

BC explanation on 20Newsgroup. Top 10 words out of 204’817

#	Baseball	Hockey	Autos	Graphics	Macintosh	Windows	Cryptography
1	Phillies	NHL	car	polygon	Centris	’AX	encryption
2	Braves	hockey	cars	TIFF	Quadra	Windows	Clipper
3	pitching	Leafs	eliot	graphics	Apple	3.1	clipper
4	Alomar	team	SHO	3D	Mac	windows	crypto
5	Baseball	Devils	automotive	3DO	Duo	W4WG	NSA
6	Players	ESPN	Callison	CView	LCIII	cica	escrow
7	Mets	Wings	Dumbest	POV	LC	font	key
8	Sox	Pens	rmt6r	cview	C650	BJ-200	DES
9	Cubs	playoffs	Thigpen	tdawson	BMUG	NDIS	Amanda
10	baseball	playoff	Toyota	MPEG	IIsi	Win	wiretap

BL explanation for the class baseball in 20Newsgroup



Code

Installation

```
> pip install bornrule
```

Scikit-Learn

```
from bornrule import BornClassifier
```

- Implements the classification algorithm
- Use it as any other `sklearn` classifier
- Supports both dense and sparse input and GPU-accelerated computing via `cupy`

PyTorch

```
from bornrule.torch import Born
```

- Implements the neural architecture
- Use it as any other `torch` layer
- Supports real and complex-valued inputs. Outputs probabilities in the range [0, 1]

SQL

```
from bornrule.sql import BornClassifierSQL
```

- Implements the classification algorithm
- Use it for in-database classification
- Supports inputs represented as json `{feature: value, ...}`

Thank you!

All code and references are available at:

`https://github.com/eguidotti/bornrule`

References

H. Shrivastava, E. Bart, B. Price, H. Dai, B. Dai, and S. Aluru.
Cooperative neural networks: Exploiting prior independence structure for improved classification.
In *Advances in Neural Information Processing Systems*, volume 31, pages 4126–4136, 2018.

I. Yamada and H. Shindo.
Neural attentive bag-of-entities model for text classification.
In *Proceedings of the 23rd Conference on Computational Natural Language Learning*, pages 563–573, 2019.

I. Yamada, H. Shindo, and Y. Takefuji.
Representation learning of entities and documents from knowledge base descriptions.
In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 190–201, 2018.

L. Yao, C. Mao, and Y. Luo.
Graph convolutional networks for text classification.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377, 2019.

S. Zhang, M. Liu, and J. Yan.
The diversified ensemble neural network.
In *Advances in Neural Information Processing Systems*, volume 33, pages 16001–16011, 2020.