An Efficient Implementation of Hubness-Aware Weighting Using Cython

Krisztian Buza

BioIntelligence Group
Department of Mathematics-Informatics
Sapientia Hungarian University of Transylvania
Targu Mures, Romania

buza@biointelligence.hu

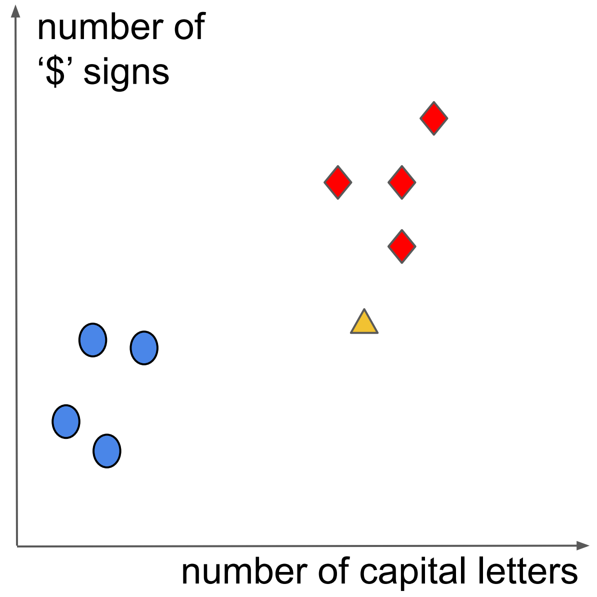
Nearest Neighbor Classification

- Simple, intuitive, explainable
- Works reasonable well with moderate amount of data

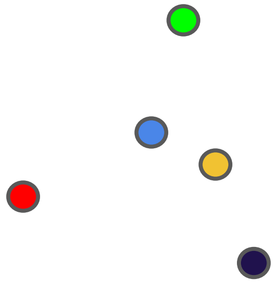
BUT

- it is affected by the detrimental effect of bad hubs

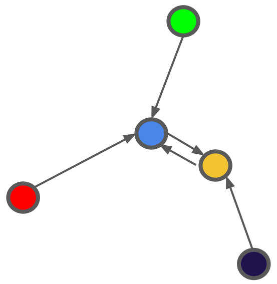
Example: spam detection



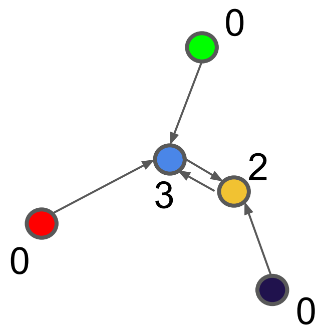
Hubness



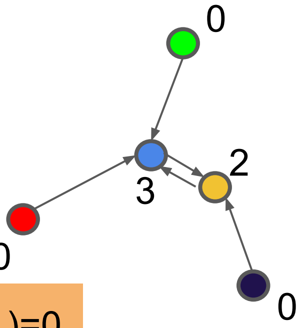
Hubness



Hubness

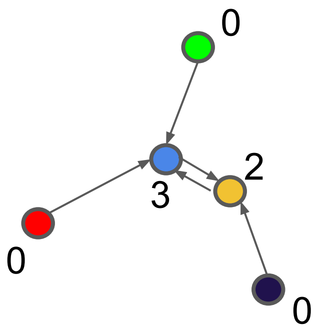


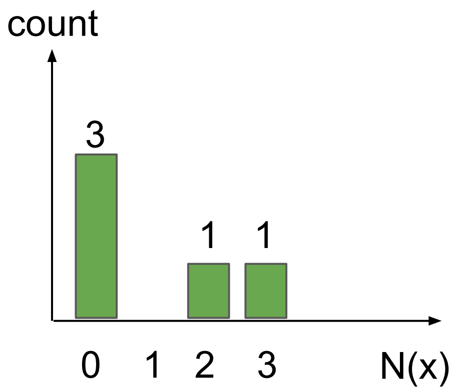
Hubness



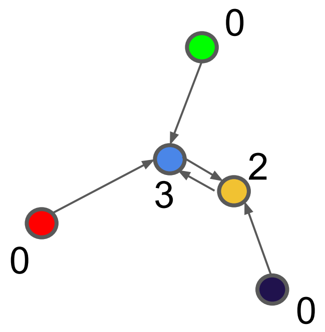
$N(x_1)=0$

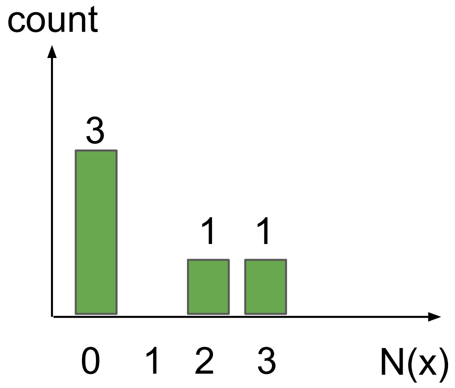
Hubness

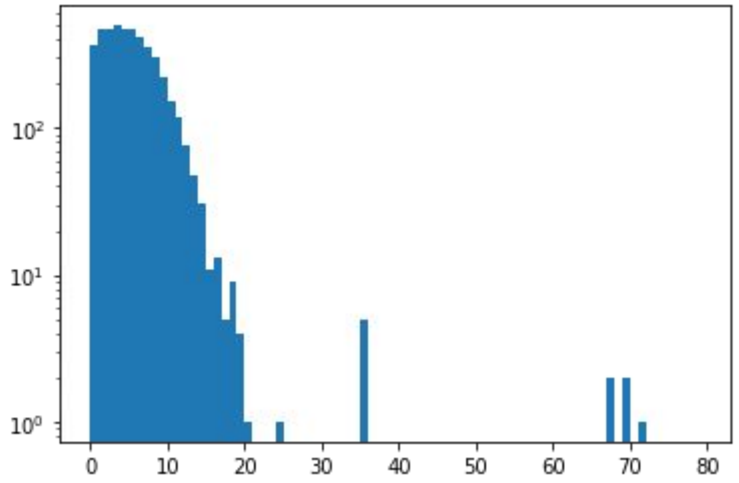




Hubness







The distribution of N(x) with k = 5 nearest neighbors in case of the Spambase dataset

Some Prominent Applications of Hubness-Aware Machine Learning Techniques

- Time series classification
- Classification of imbalanced data
- Clustering
- Collaborative Filtering
- Classification of gene expression data
- Drug-target interaction prediction
- Person identification based on keystroke dynamics
- Hubness-aware ensembles
- Hubness-aware weighting for neural networks
- …

Hubness-Aware Weighting

- an instance x is a bad neighbor of another instance x' if x is one of the k-nearest neighbors of x' and their class labels are different
- $BN_k(x)$ = how many times an instance x appears as bad neighbor of other instances
- normalized bad hubness score:

$$h_b(x) = \frac{BN_k(x) - \mu(BN_k)}{\sigma(BN_k)}$$

where

$\mu(BN_k)$ = mean of $BN_k(x)$

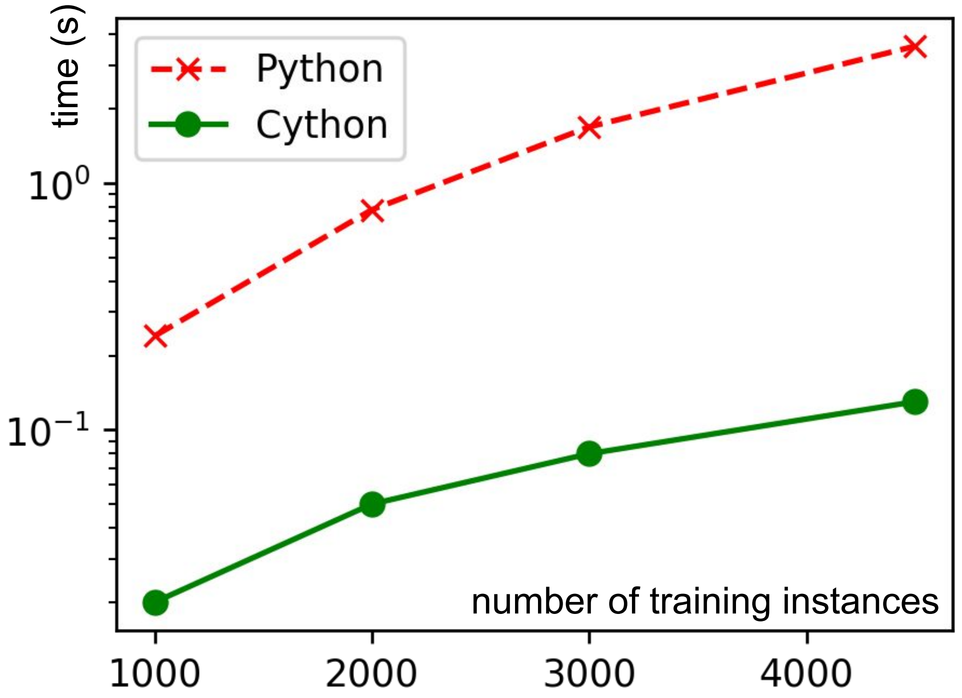
$\sigma(BN_k)$ = standard deviation of $BN_k(x)$

- weighted k-nearest neighbor classification, weights: $w(x) = e^{-h_b(x)}$

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. 2009. Nearest neighbors in high-dimensional data: The emergence and influence of hubs. In Proceedings of the 26th Annual International Conference on Machine Learning. 865–872.

Experiments

- We implemented hubness-aware weighting both in Python and Cython
- Experiments on the Spambase dataset
- The Cython-based version is much faster while both have the exactly same accuracy



Conclusions & Outlook

- Implementation of computationally expensive functions in Cython may speed up various calculations (code is compiled, less time is needed for type inference when the code is executed)

- Try out our code yourself: https://github.com/kr7/cython/