# News Article Classification
## (Multi-Label Learning: ML-KNN & BP-MLL)

Lauren Contard, Archit Datar, Bobby Lumpkin, Haihang Wu

The Ohio State University

STAT 6500

# Overview

# Introduction and Problem Statement

# KNN Based Approaches

# ML-KNN Algorithm: Overall Approach

**Overall Approach:** This ML-KNN algorithm takes a parametric, Bayesian approach towards estimating the Bayes Optimal Classifier. As with the single-label algorithm, it does this using the K nearest neighbors of an instance. Namely...

# ML-KNN Algorithm: Overall Approach

**Overall Approach:** This ML-KNN algorithm takes a parametric, Bayesian approach towards estimating the Bayes Optimal Classifier. As with the single-label algorithm, it does this using the K nearest neighbors of an instance. Namely...

- Given a test instance, $t$, $\vec{Y}_t$ is determined using the MAP estimate:

# ML-KNN Algorithm: Overall Approach

**Overall Approach:** This ML-KNN algorithm takes a parametric, Bayesian approach towards estimating the Bayes Optimal Classifier. As with the single-label algorithm, it does this using the K nearest neighbors of an instance. Namely...

- Given a test instance, $t$, $\vec{Y}_t$ is determined using the MAP estimate:

$$\begin{aligned}
\vec{y}_t(\ell) &= \underset{b\in\{0,1\}}{\operatorname{argmax}} \, \mathbb{P}\left(\mathrm{H}_b^\ell | E^\ell_{\vec{C}_t(\ell)}\right), \quad \ell \in \mathcal{Y} \\
&= \underset{b\in\{0,1\}}{\operatorname{argmax}} \, \frac{\mathbb{P}\left(\mathrm{H}_b^\ell\right) \cdot \mathbb{P}\left(E^\ell_{\vec{C}_{t(\ell)}} | \mathrm{H}_b^\ell\right)}{\mathbb{P}\left(E^\ell_{\vec{C}_t(\ell)}\right)} \\
&= \underset{b\in\{0,1\}}{\operatorname{argmax}} \, \mathbb{P}\left(\mathrm{H}_b^\ell\right) \cdot \mathbb{P}\left(E^\ell_{\vec{C}_{t(\ell)}} | \mathrm{H}_b^\ell\right)
\end{aligned}$$

# ML-KNN Algorithm: Overall Approach

**Overall Approach:** This ML-KNN algorithm takes a parametric, Bayesian approach towards estimating the Bayes Optimal Classifier. As with the single-label algorithm, it does this using the K nearest neighbors of an instance. Namely...

- Given a test instance, $t$, $\vec{Y}_t$ is determined using the MAP estimate:

$$\begin{aligned}
\vec{y}_t(\ell) &= \underset{b\in\{0,1\}}{\text{argmax}}\, \mathbb{P}\left(\mathrm{H}_b^\ell | E^\ell_{\vec{C}_t(\ell)}\right), \quad \ell \in \mathcal{Y} \\
&= \underset{b\in\{0,1\}}{\text{argmax}}\, \frac{\mathbb{P}\left(\mathrm{H}_b^\ell\right) \cdot \mathbb{P}\left(E^\ell_{\vec{C}_{t(\ell)}} | \mathrm{H}_b^\ell\right)}{\mathbb{P}\left(E^\ell_{\vec{C}_t(\ell)}\right)} \\
&= \underset{b\in\{0,1\}}{\text{argmax}}\, \mathbb{P}\left(\mathrm{H}_b^\ell\right) \cdot \mathbb{P}\left(E^\ell_{\vec{C}_{t(\ell)}} | \mathrm{H}_b^\ell\right)
\end{aligned}$$

- Where we take a Bayesian approach towards estimating the prior probabilities, $\mathbb{P}\left(\mathrm{H}_b^\ell\right)$, and conditional probabilities, $\mathbb{P}\left(E^\ell_{\vec{C}_{t(\ell)}} | \mathrm{H}_b^\ell\right)$.

# ML-KNN Algorithm: More Notation

**Notation:**

# ML-KNN Algorithm: More Notation

**Notation:**

- Let $N(x)$ denote the set of $K$ nearest neighbors of $x$, identified in the training set.

# ML-KNN Algorithm: More Notation

**Notation:**

- Let $N(x)$ denote the set of $K$ nearest neighbors of $x$, identified in the training set.
- Let $\vec{C}_x(\ell) = \sum_{a \in N(x)} \vec{y}_a(\ell)$ $(\ell \in \mathcal{Y})$ define a membership counting vector.

**Notation:**

- Let $N(x)$ denote the set of $K$ nearest neighbors of $x$, identified in the training set.
- Let $\vec{C}_x(\ell) = \sum_{a \in N(x)} \vec{y}_a(\ell)$ $(\ell \in \mathcal{Y})$ define a membership counting vector.
- Let $H_0^\ell$ denote the event that test instance $t$ does not have a label $\ell$ and let $H_1^\ell$ denote the event that it does have label $\ell$.

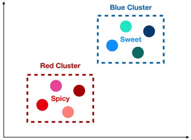
# ML-KNN Algorithm: More Notation

**Notation:**

- Let $N(x)$ denote the set of $K$ nearest neighbors of $x$, identified in the training set.
- Let $\vec{C}_x(\ell) = \sum_{a \in N(x)} \vec{y}_a(\ell)$ $(\ell \in \mathcal{Y})$ define a membership counting vector.
- Let $H_0^\ell$ denote the event that test instance $t$ does not have a label $\ell$ and let $H_1^\ell$ denote the event that it does have label $\ell$.
- Let $E_j^\ell$ $(j \in \{1, ..., K\})$ denote the event that, among the $K$ nearest neighbors of $t$, there are exactly $j$ instances which have label $\ell$.

# ML-KNN Algorithm: Reasons for dimension reduction

Data points




Low dimensional space

| | Reddish | Bluish |
|---|---|---|
| | 1 | 0 |
| | 1 | 0 |
| | 1 | 0 |
| | 1 | 0 |
| | 0 | 1 |
| | 0 | 1 |
| | 0 | 1 |
| | 0 | 1 |

High dimensional space

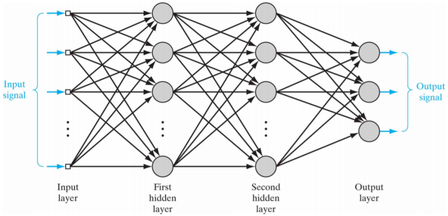
| | Red | Maroon | Pink | Flamingo | Blue | Turquoise | Seaweed | Ocean |
|---|---|---|---|---|---|---|---|---|
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Having a high dimensional feature space causes Euclidian distances between points to be fairly similar as the distance vector components are partitioned across many dimensions.

# Neural Network Based Approaches

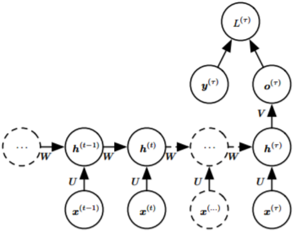
## Feed Forward Networks

- Neurons in the first layer represent components of the input vectors.
- The output of the neuron in the next layer is determined by applying a non-linear "activation function" to a linear combination of the input components, plus a bias.



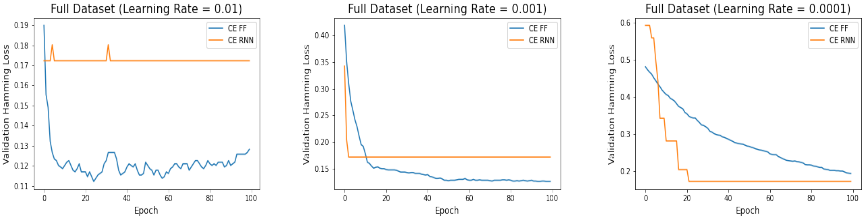

## Recurrent Neural Networks (RNNs)

- RNNs are a popular adaptation for NLP problems because they are uniquely suited for sequence processing.
- This is due to their hidden unit connections with shared weights, which allow for information from previous and/or future states to influence the current state.



# Naive vs Novel Approaches (Cross Entropy vs BPMLL)

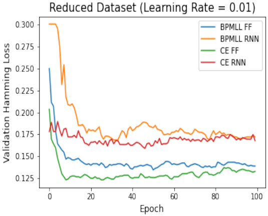
# Artificial Neural Network Results: Full Dataset

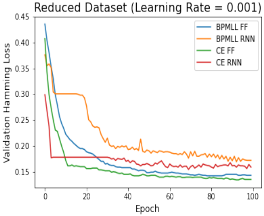


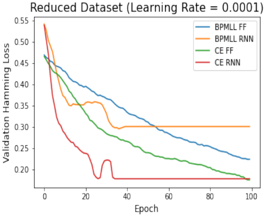

| Learning Rate | CE FF | CE RNN |
|---|---|---|
| 0.01 | 0.3020833333 | 0.2379807692 |
| 0.001 | 0.17868589740 | 0.1730769231 |
| 0.0001 | 0.2115384615 | 0.1770833333 |

Table 1: Hamming Loss with Threshold Function Learning

CE FF outperform CE RNN in constant threshold but underperform in learned threshold; The effect of learning rate.




| Learning Rate | CE FF | BPMLL FF | CE RNN | BPMLL RNN |
|---|---|---|---|---|
| 0.01 | 0.1520979021 | 0.145979021 | 0.2447552448 | 0.2194055944 |
| 0.001 | 0.1853146853 | 0.2578671329 | 0.1791958042 | 0.20454545 |
| 0.0001 | 0.2071678322 | 0.1844405594 | 0.1896853147 | 0.2132867133 |

Table 2: Hamming Loss with Threshold Function Learning

Same conclusion for RNN and FF; BPMLL shows NO better performance in hamming loss than cross entropy.

# Discussion and Conclusions

# References

Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007. doi: 10.1016/j.patcog.2006.12.019.

Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006. doi: doi:10.1109/TKDE.2006.162.