Efficient Loss-Based Decoding on Graphs for Extreme Classification

Itay Evron, Edward Moroshko, and Koby Crammer

TECHNION Israel Institute of Technology

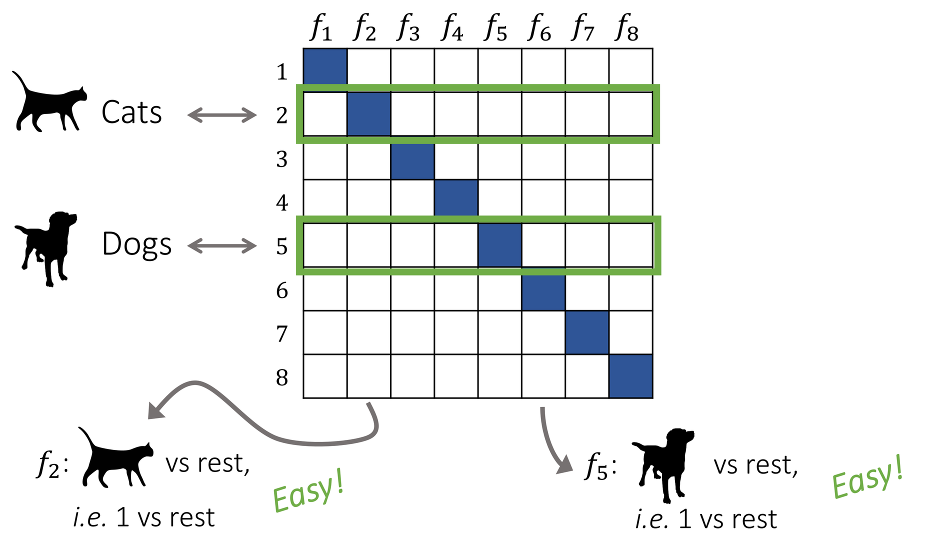
Extreme multiclass classification

- Tasks with an extremely large number of classes K.
  - Time and space complexities during training and inference become critical.
- **We propose a graph-based classification scheme with time and space complexities logarithmic in K.**

One vs Rest – Simple but expensive

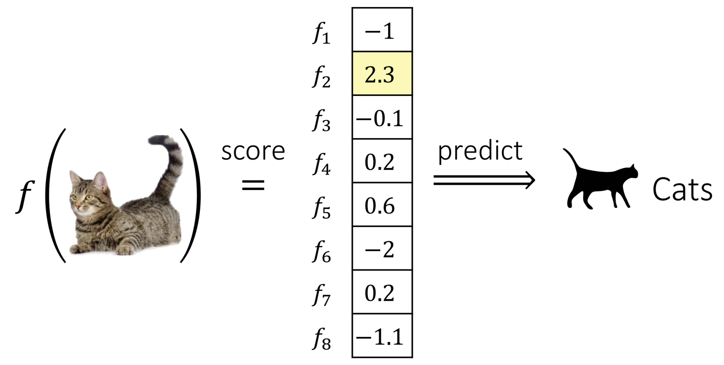
Training:

- Learn K binary classifiers $f_1, \dots, f_K : \mathcal{X} \to \mathbb{R}$



Inference for $x \in \mathcal{X}$:

- Score all: $f(x) = [f_1(x), \dots, f_K(x)]$
- Predict: $\hat{y} = \arg\max_k f_k(x)$

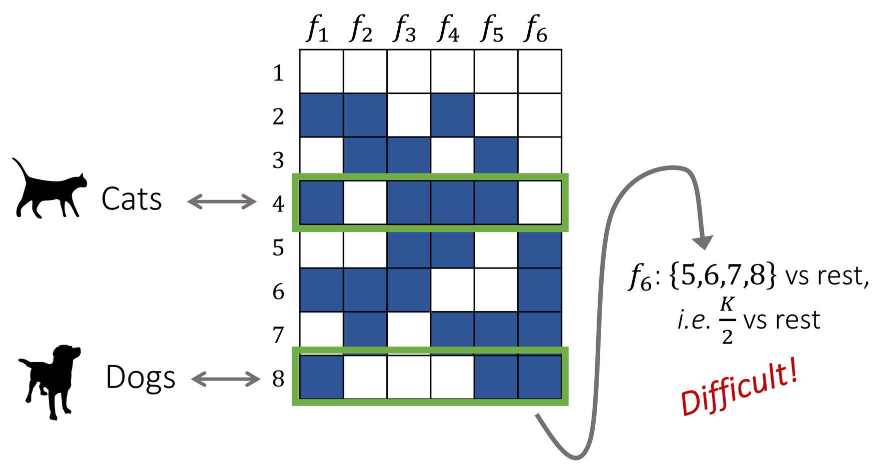


Model size (#classifiers) $O(K)$ | Inference time $O(K)$

Error Correcting Output Coding

Training:

- Get a coding matrix $M \in \{-1,1\}^{K \times \ell}$
- Map each codeword to a class (e.g. arbitrarily)
- Learn ℓ binary classifiers $f_1, \dots, f_\ell : \mathcal{X} \to \mathbb{R}$

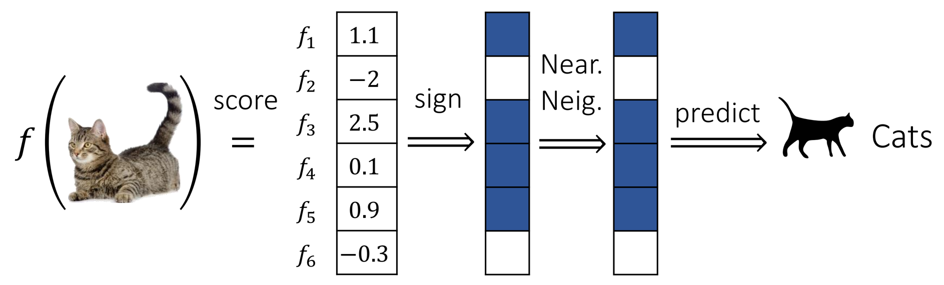


Model size (#classifiers) $O(\ell) = O(\log K)$ | Inference time $O(K\ell) = O(K \log K)$

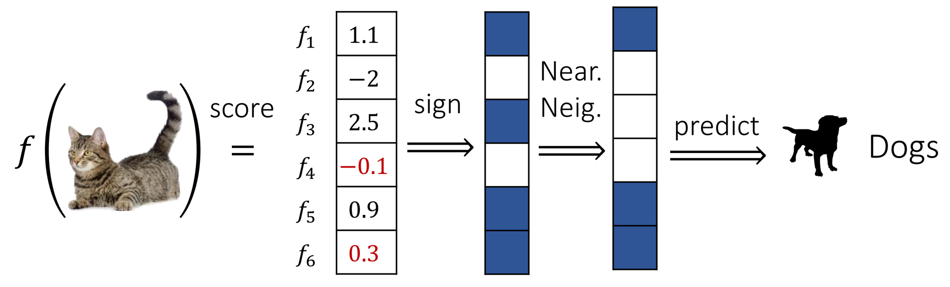
Inference for $x \in \mathcal{X}$:

- Score all: $f(x) = [f_1(x), \dots, f_\ell(x)]$
- Predict: $\hat{y} = \arg\min_k d_{Hamm}\left(M_k,\ sign(f(x))\right)$

Inference example:



Mistake example:



Loss based decoding

- Instead of minimizing the Hamming distance, predict the class that minimizes some loss:

$$\hat{y} = \arg\min_k \sum_{j=1}^{\ell} \mathcal{L}\left(M_{k,j} \times f_j(x)\right)$$

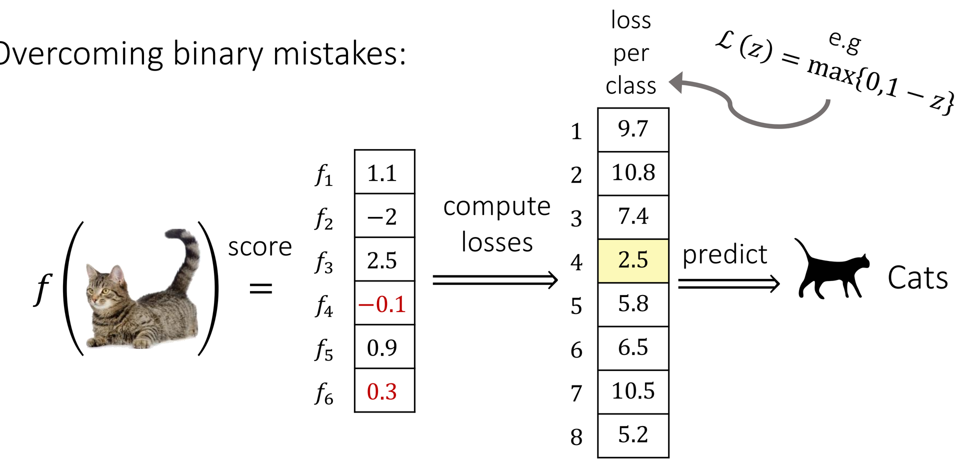
- An upper bound of the training multiclass error is proportional to:

$$\frac{\ell \times \varepsilon}{\rho}$$

Number of classifiers — ℓ; Average binary loss — ε; Minimum row distance — ρ

- The decoding loss function matters.

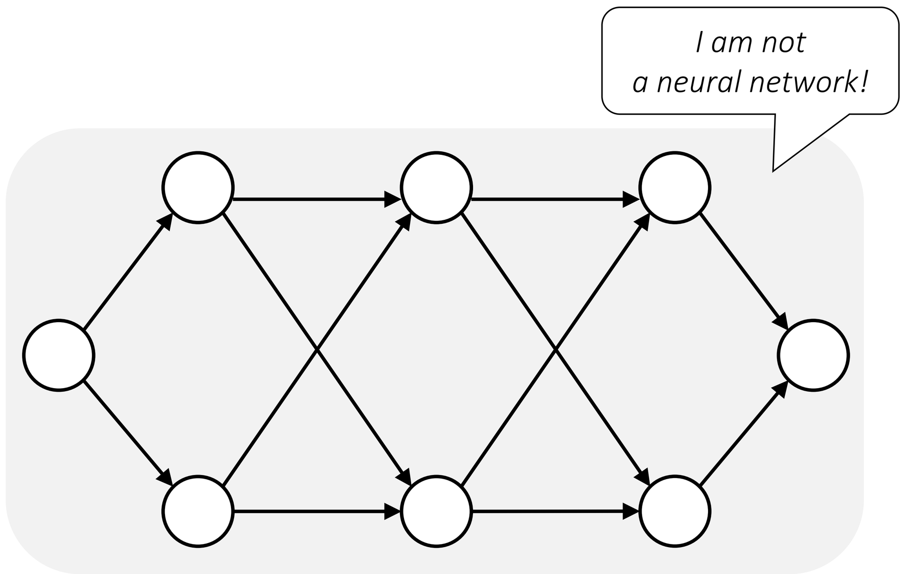
Overcoming binary mistakes:



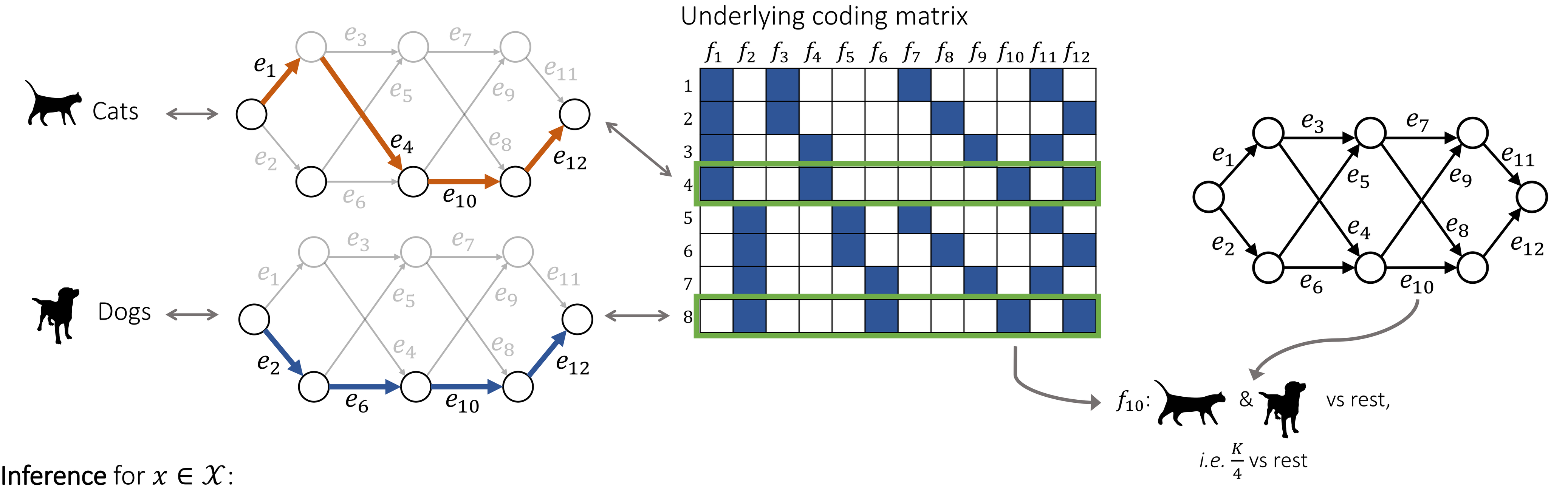
Inference time $O(K\ell) = O(K \log K)$

Our model – Wide-LTLS

- Based on LTLS *[Jasinska and Karampatziakis 2016]*.
- Build a trellis graph with exactly K paths.
- Map each path to a class (e.g. arbitrarily).

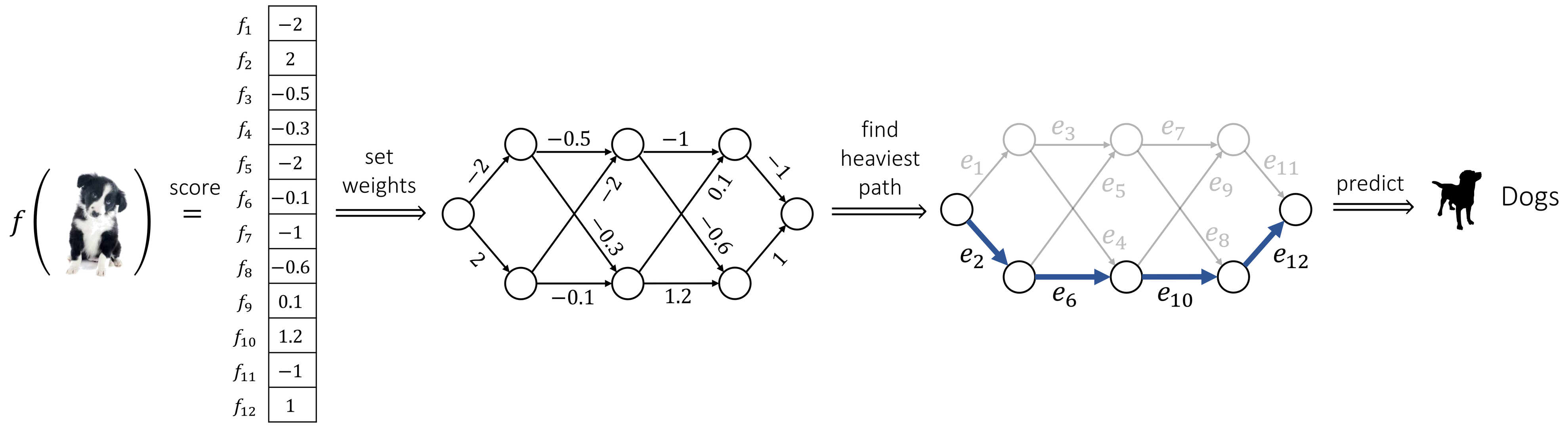


- For each edge e *(in parallel)*:
  - **Train** a binary classifier $f_e : \mathcal{X} \to \mathbb{R}$ on the entire training set:
    - Separate classes (=paths) that use this edge, from the classes that do not.



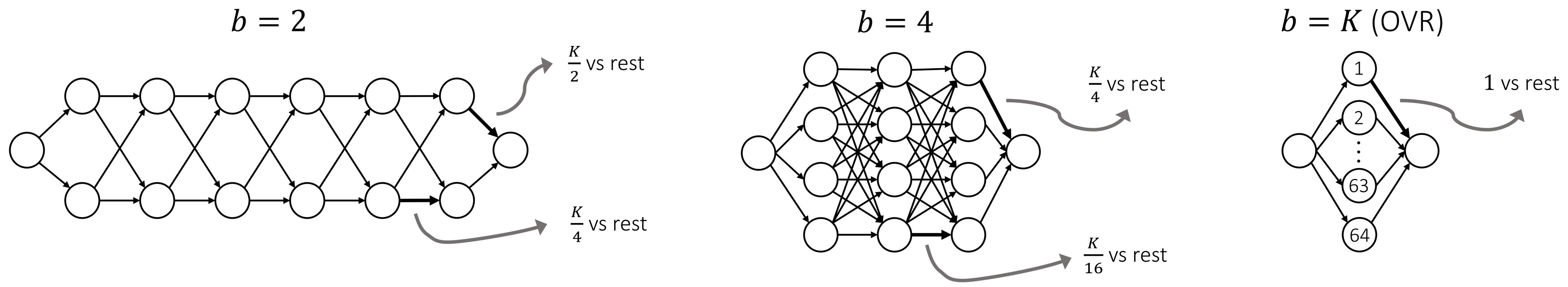
Inference for $x \in \mathcal{X}$:

- Set all edge weights: $w(e) = f_e(x)$.
- Find the heaviest path and predict its class.



Increasing the graph width

- The same number of classes can be represented with different graph widths b.
- For instance, the following graphs all have $K = 64$ paths (=classes):

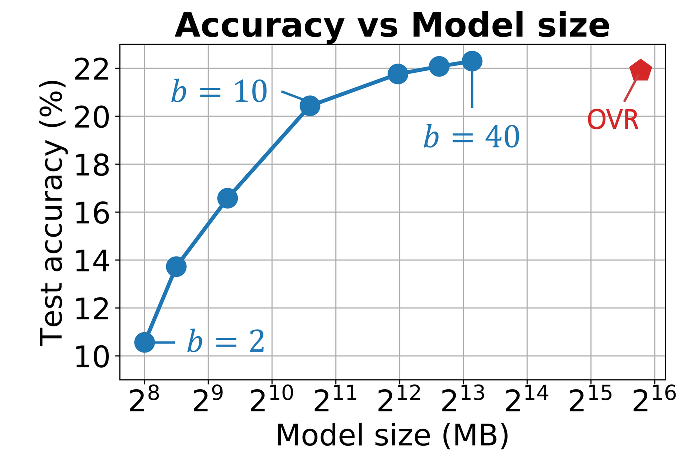


Model size (#classifiers) $|E| = O\left(\frac{b^2}{\log b} \cdot \log K\right)$ | Inference time (for $b < \sqrt{K}$) $O\left(\frac{b^2}{\log b} \cdot \log K\right)$

Graph width controls performance

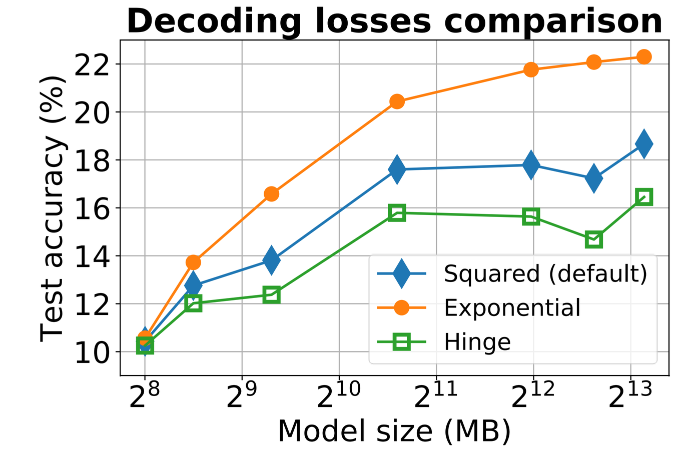
- Our model offers a tradeoff between accuracy and model size.

LSHTC1 dataset
$K = 12{,}294$
$d = 1{,}199{,}856$
$d_{nnz} = 172$



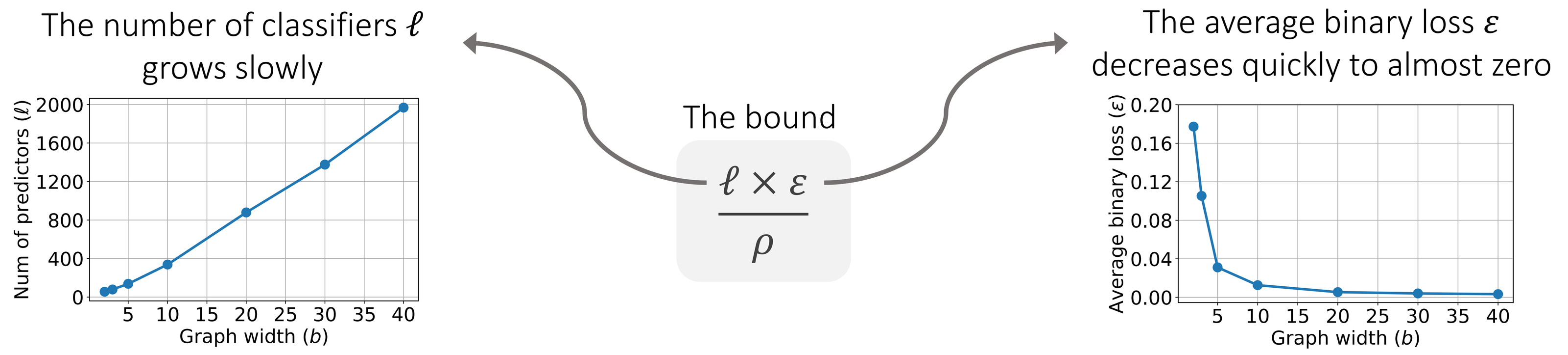
W-LTLS as loss-based decoding

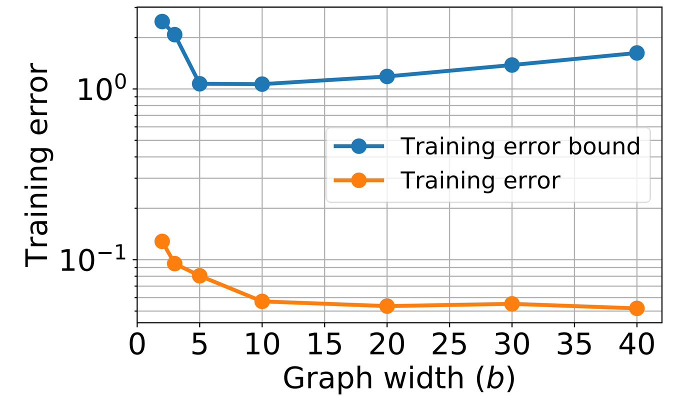
- We prove that W-LTLS performs loss-based decoding with the squared loss $\mathcal{L}(z) = (1-z)^2$.
- We show how to generalize W-LTLS to any loss function $\mathcal{L}$, and perform loss based decoding in time logarithmic in K.
- The decoding loss function matters!
  - The loss function can be chosen quickly after training.



Wider graph – Easier binary problems

- The subproblems are $\frac{K}{b^2}$ vs rest, thus get easier.





https://github.com/ievron/wltls/ , arXiv: 1803.03319