Efficient Loss-Based Decoding on Graphs for Extreme Classification

Itay Evron, Edward Moroshko, and Koby Crammer

TECHNION
Israel Institute
of Technology

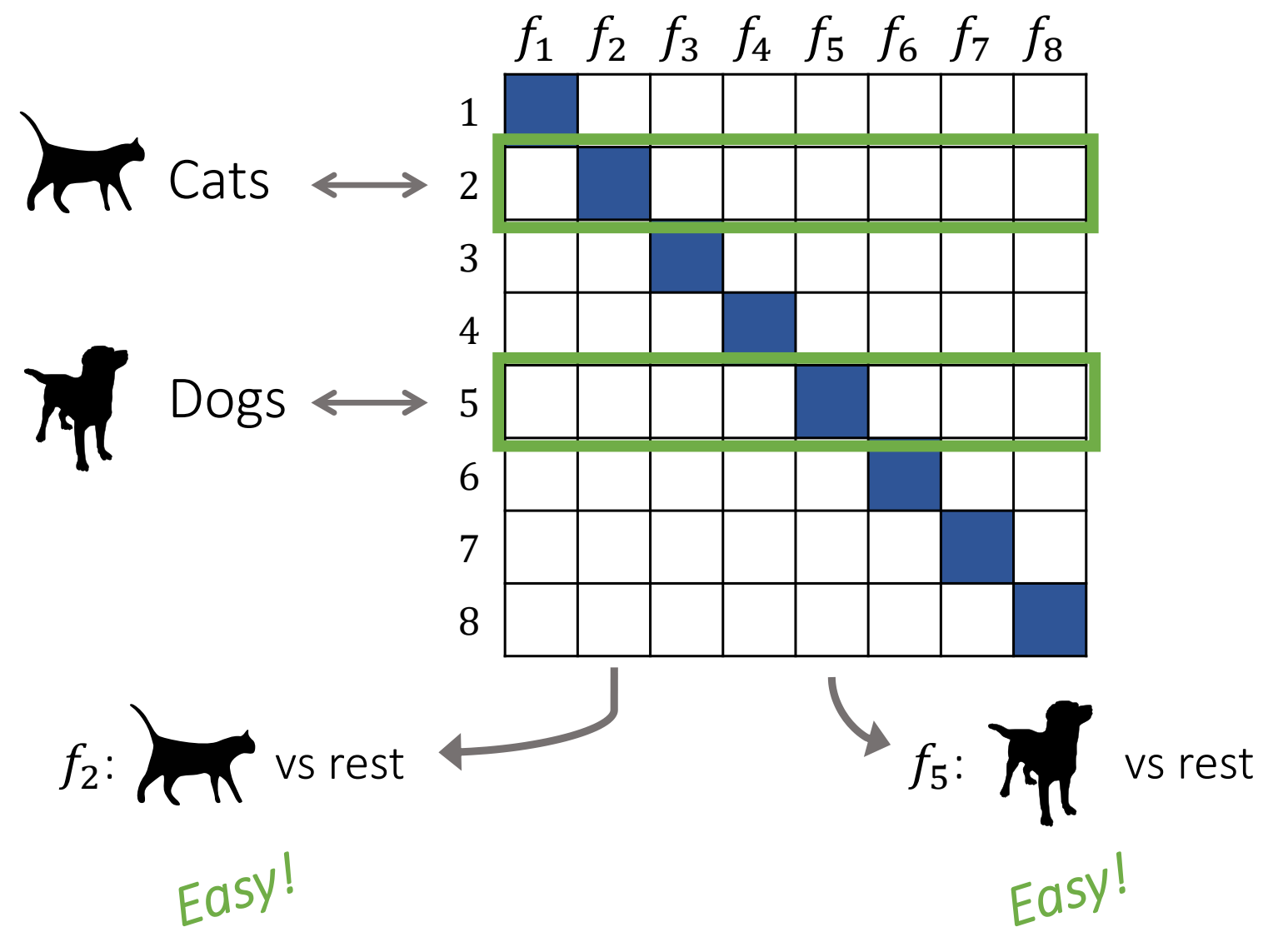
Extreme multiclass classification

- Tasks with an extremely large number of classes K.
  - Time and space complexities during training and inference become critical.
  - Datasets are typically sparse, i.e. samples have on average only $d_{nnz} \ll d$ nonzero features.
- **We propose a graph-based classification scheme with time and space complexities logarithmic in $\mathbf{K}$.**

One vs Rest – Simple but expensive

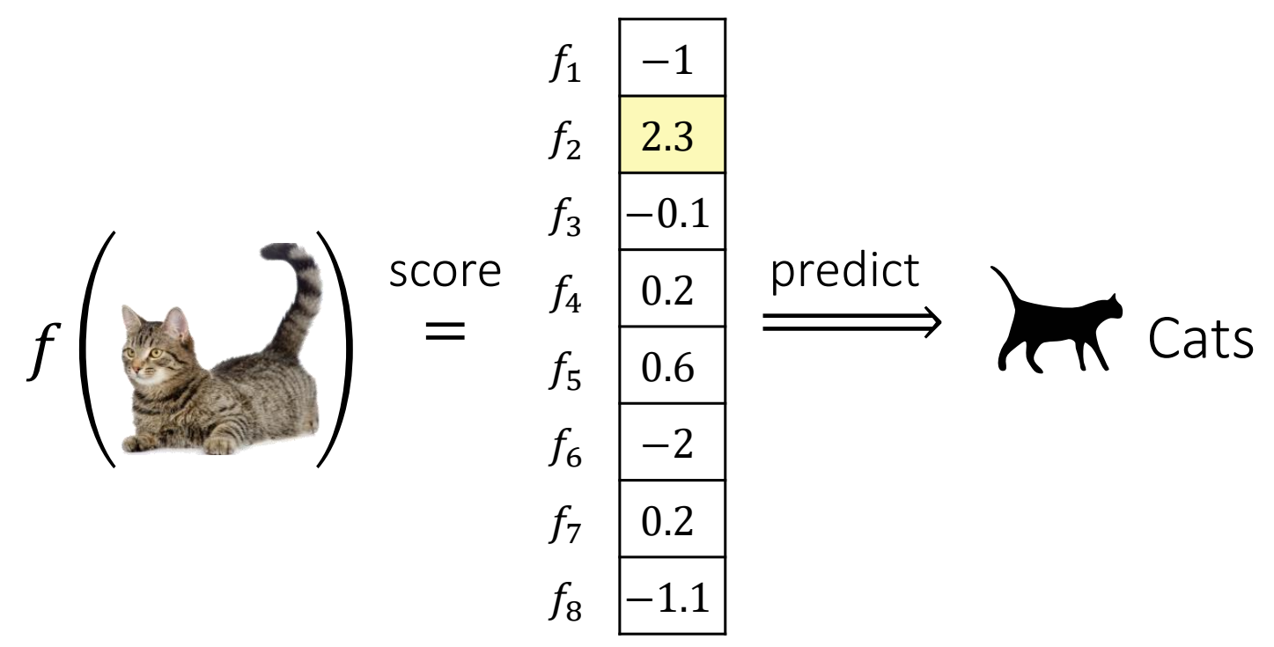
Training:

- 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: $\hat{y} = \arg\max_k f_k(x)$



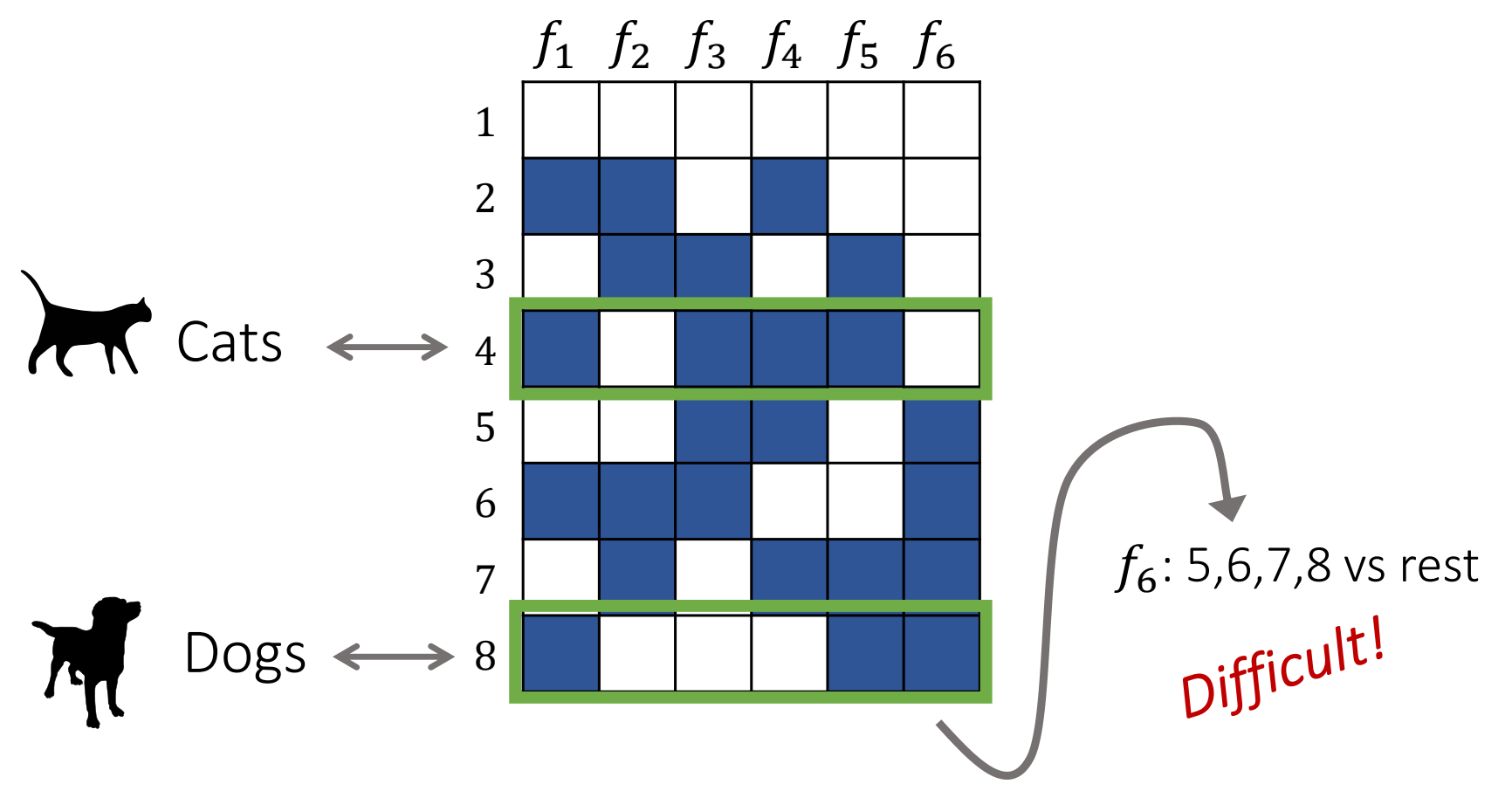
Model size $O(dK)$

Inference time $O(d_{nnz}K)$

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 $f_1, \dots, f_\ell : \mathcal{X} \to \mathbb{R}$



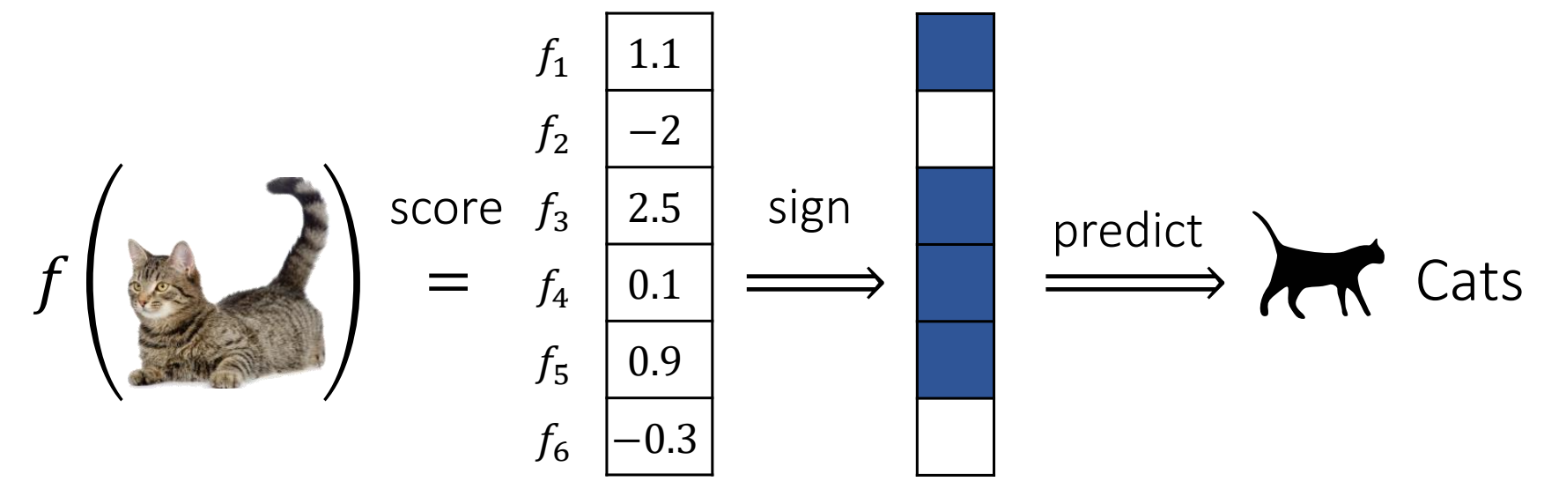
Model size $O(\underbrace{d\ell}_{\text{linear classifiers}} + \underbrace{K\ell}_{\text{coding matrix}})$

Inference time $O(\underbrace{d_{nnz}\ell}_{\text{score}} + \underbrace{K\ell}_{\text{find best codeword}})$

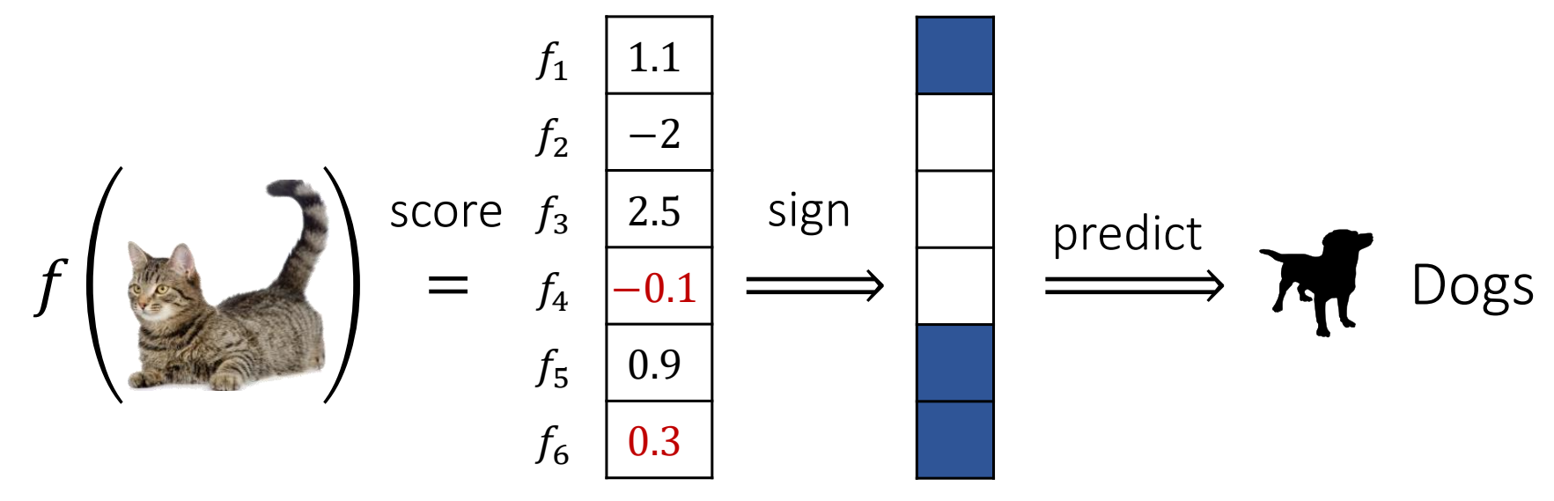
Inference for $x \in \mathcal{X}$:

- 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,

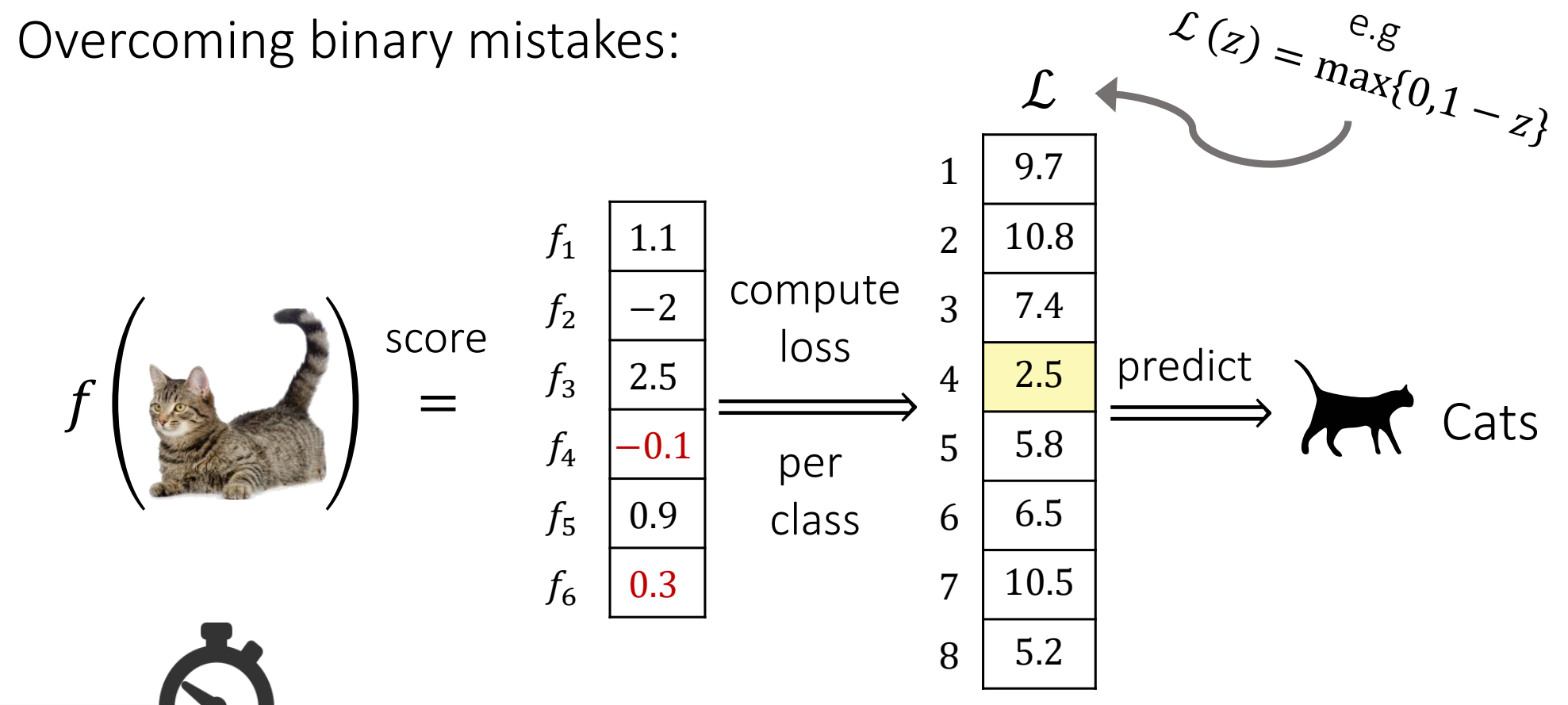
  predict $\hat{y} = \arg\min_k \sum_{j=1}^{\ell} \mathcal{L}\left(M_{k,j} \times f_j(x)\right)$
- An upper bound of the training multiclass error is proportional to:

Number of predictors — ℓ × Average binary loss — ε
Minimum row distance — ρ

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

- The decoding loss function matters.

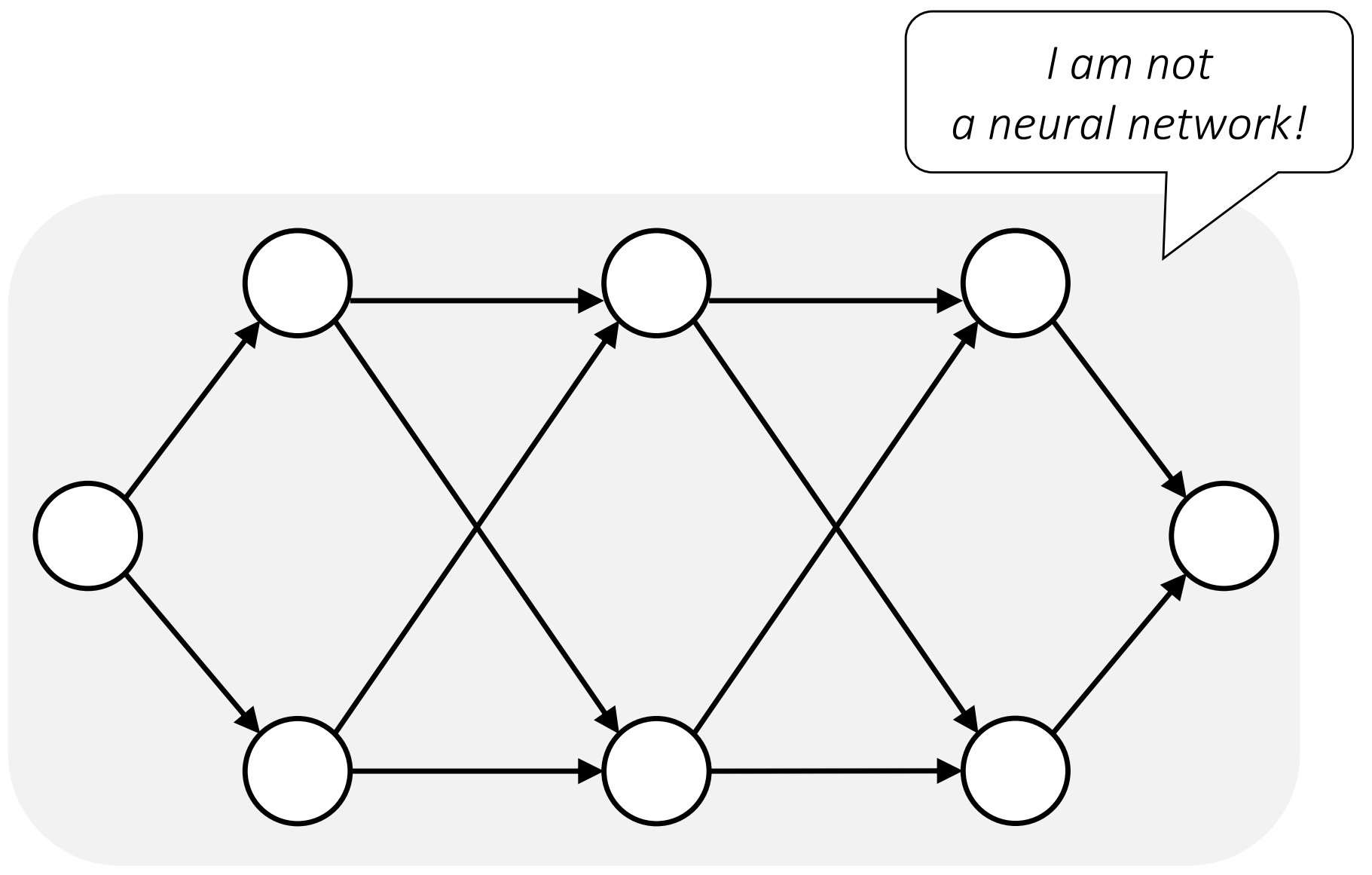
Overcoming binary mistakes:

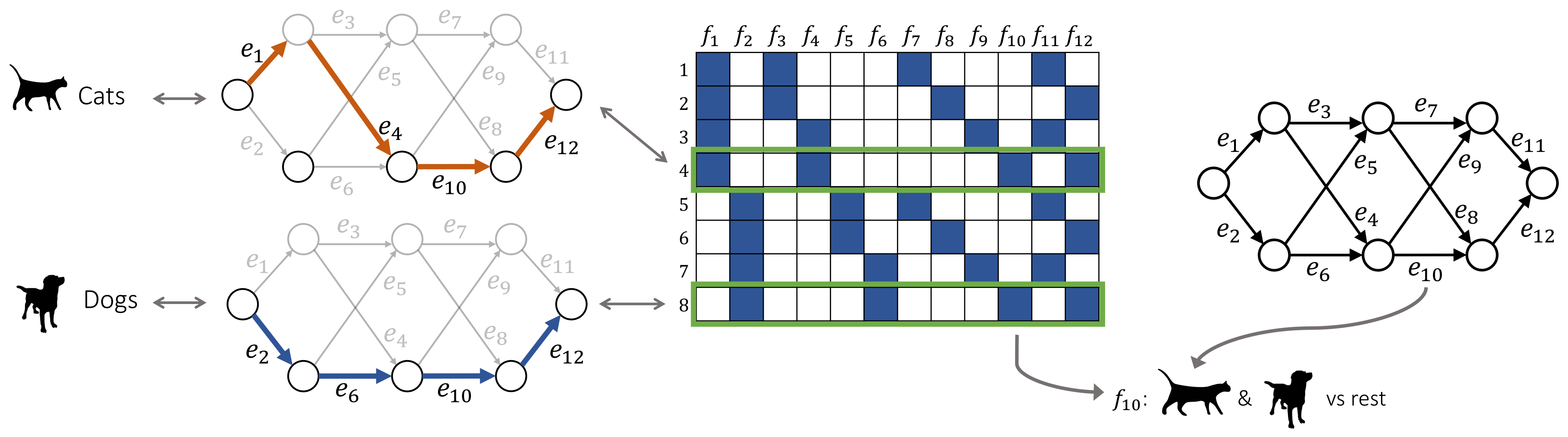


Inference time $O(\underbrace{d_{nnz}\ell}_{\text{score}} + \underbrace{K\ell}_{\text{loss based decoding}})$

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 $e \in E$:
  - **Train** a 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.

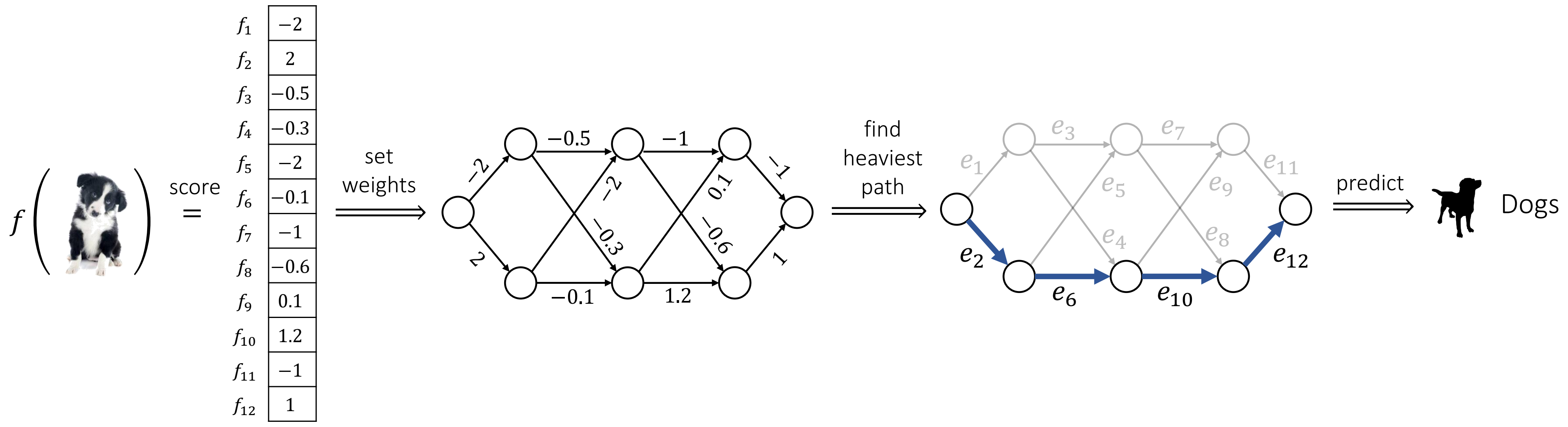




The binary problems are $\frac{1}{4}K$ vs rest

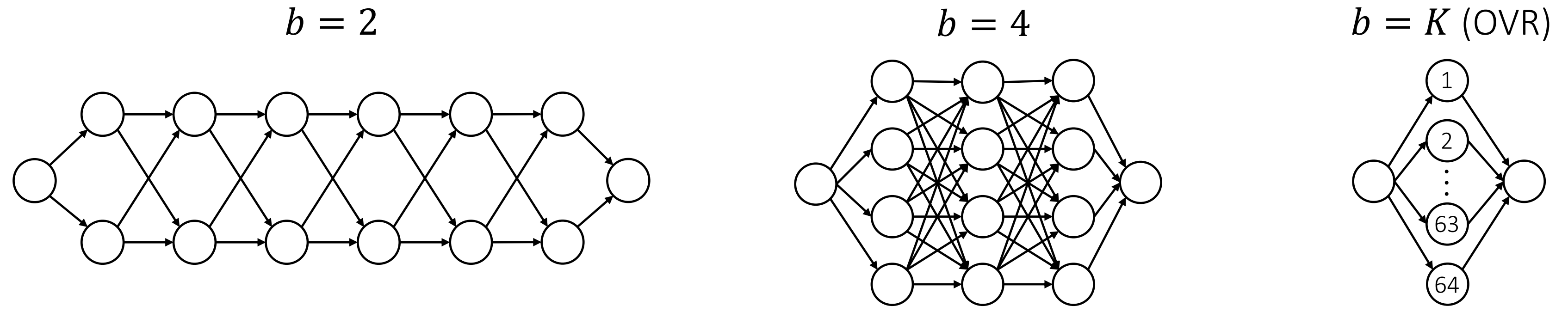
Inference for x $\in \mathcal{X}$:

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



Graph width controls complexity

- The following graphs have $K = 64$ paths (=classes), but different graph widths b



- The number of edges / classifiers is: $|E| = O\left(\frac{b^2}{\log b} \cdot \log K\right)$.
- Therefore,

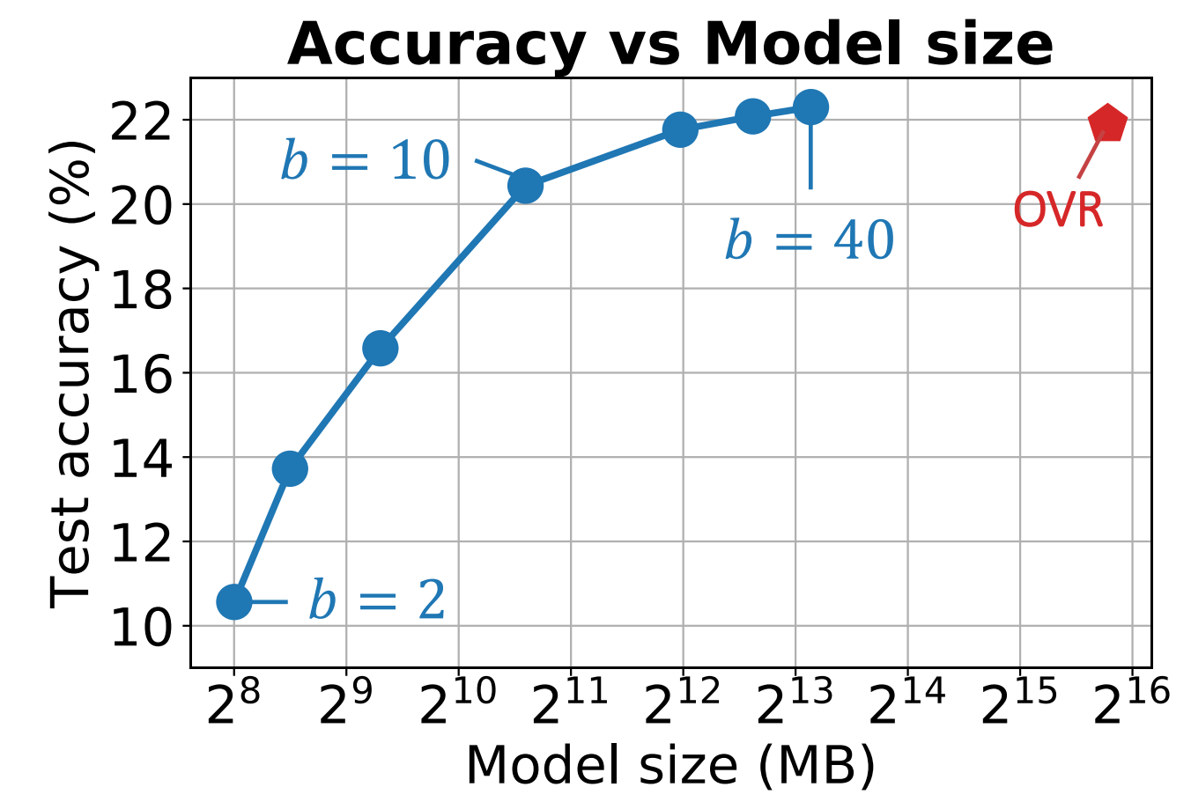
Model size $O\left(d \cdot \frac{b^2}{\log b} \cdot \log K\right)$

Inference time (for $b < \sqrt{K}$) $O\left(d_{nnz} \cdot \frac{b^2}{\log b} \cdot \log K\right)$

Graph width controls performance

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

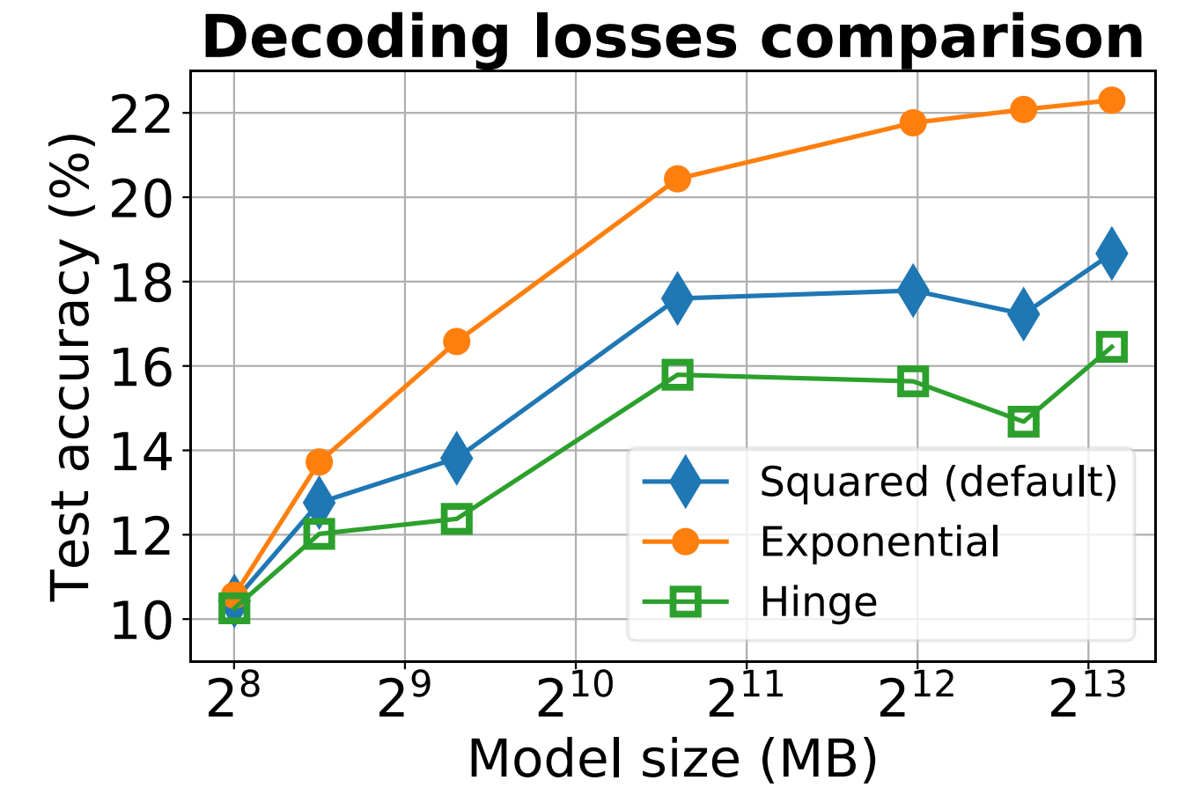
LSHTC1 dataset
$K = 12{,}294$
$d = 1{,}199{,}856$
$d_{nnz} = 172$



But why?

W-LTLS as loss-based decoding

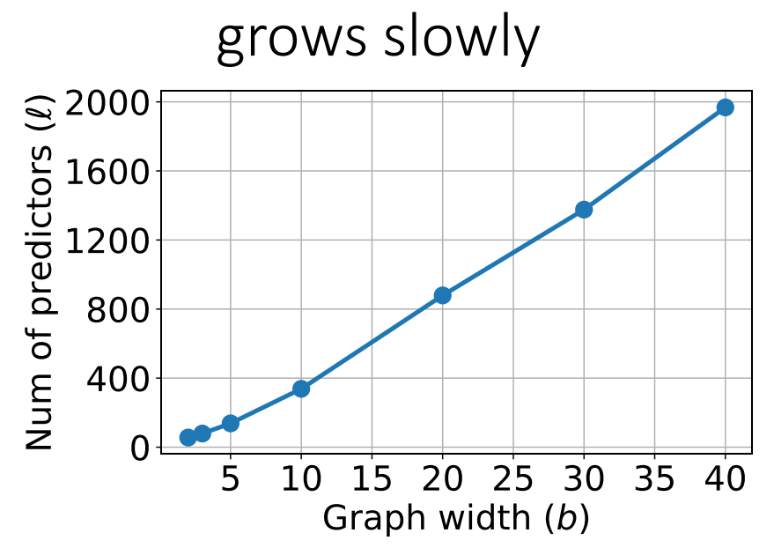
- We prove that W-LTLS performs loss-based decoding with the squared loss $\mathcal{L}(z) = (1-z)^2$.
- We show how to generalize W-LTLS to any loss function $\mathcal{L}$, and perform loss based decoding in time logarithmic in K.
- The decoding loss function matters!
  - The loss function can be chosen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



Wider graph – Easier binary problems

- The subproblems are $\frac{1}{b^2}K$-vs-rest, thus get easier.

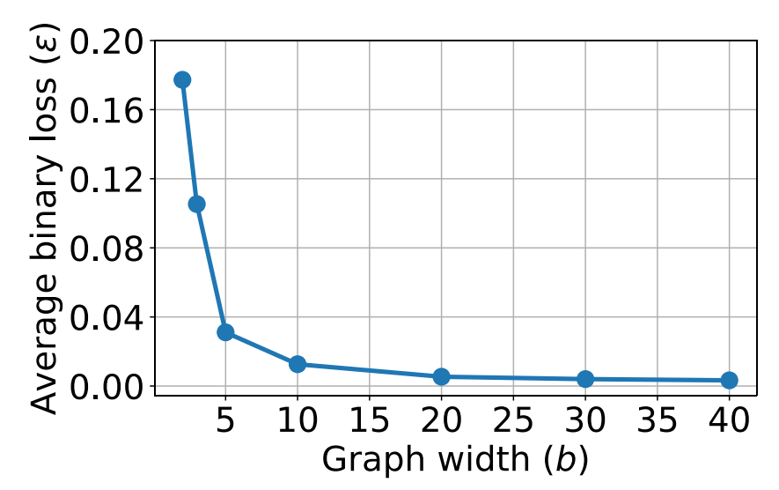
The predictor number ℓ grows slowly



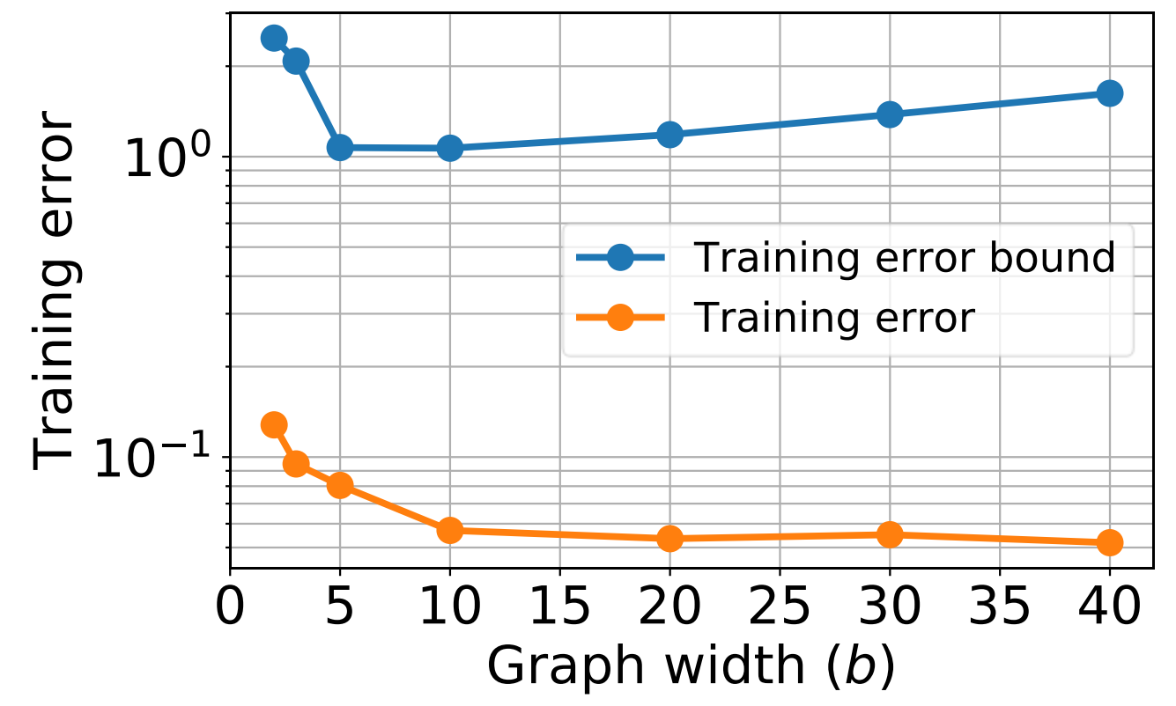
The bound

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

The average binary loss ε decreases quickly to almost zero



The bound can help find a sweet spot



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