

How to Select Pre-Trained Code Models for Reuse? A Learning Perspective

Zhangqian Bi^{1,2}, Yao Wan^{1,2,*}, Zhaoyang Chu^{1,2}, Yufei Hu², Junyi Zhang², Hongyu Zhang³,
Guandong Xu⁴, Hai Jin^{1,2}

¹*National Engineering Research Center for Big Data Technology and System,
Services Computing Technology and System Lab, Cluster and Grid Computing Lab, Wuhan, China*

²*School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China*

³*School of Big Data and Software Engineering, Chongqing University, Chongqing, China*

⁴*School of Computer Science, University of Technology Sydney, Sydney, Australia*

The 32th IEEE International Conference on Software Analysis, Evolution and Reengineering
(SANER2025)

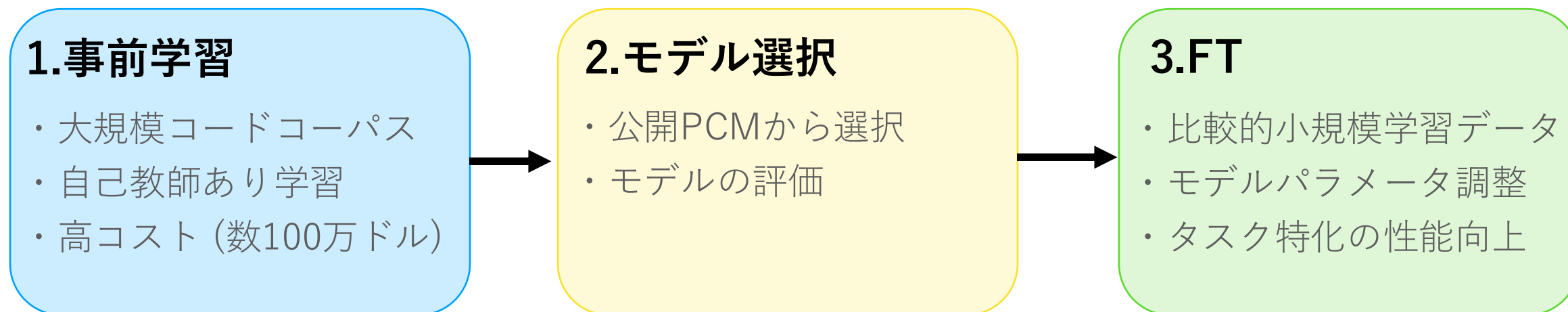
✂arXiv Preprint

背景 | 事前学習済みコードモデル

近年、事前学習済みコードモデル（PCM）が急増

- ・ コード生成・要約、脆弱性検出など、コード関連タスクに有用
 - 例：CodeBERT, CodeT5, Code Llamaなど

PCMは事前学習→ファインチューニングの流れで利用される

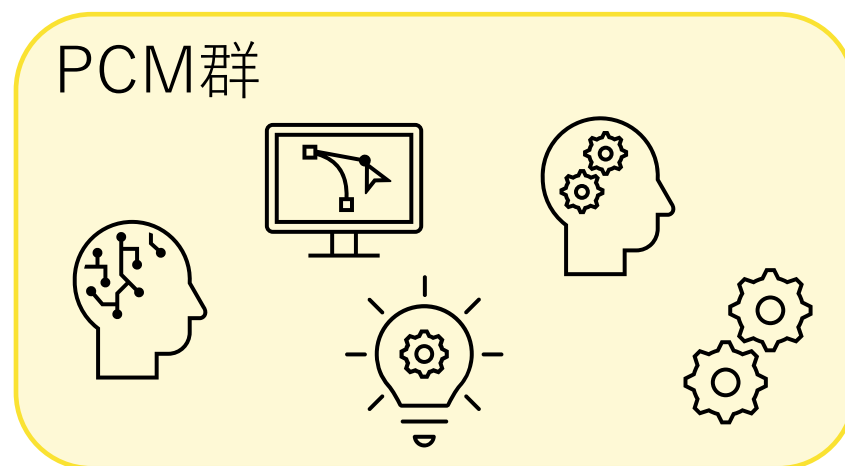


- 事前学習には膨大な計算コストがかかるため、既存のPCMを選択・利用するのが一般的

背景 | PCMの選択

課題：特定のタスクに向けた最適なモデル選択が困難

- 既存のPCMは、数が多いだけでなく、
- アーキテクチャ、サイズ、事前学習データがそれぞれ異なり、単純比較が難しい



どれを選ぶべきかわからない...

計算リソースが限られている環境では、効率的なモデル選択が特に重要

- 学術機関や中小企業など
- 適切でないモデル選択は、無駄な計算リソースの消費や性能低下につながる

問題 | 既存のモデル選択方法

①モデルサイズに基づく選択

- より大きいモデルほど性能が良いという仮定

②事前学習データに基づく選択

- データ量や, タスクの類似性で判断

③総当たりファインチューニングによる選択

- すべてのモデルを実際にファインチューニングして性能を比較

問題 | 既存のモデル選択方法

①モデルサイズに基づく選択

- より大きいモデルほど性能が良いと

モデルサイズが
大きいほど性能が良い
とは限らない

②事前学習データに基づく選択

- データ量や、タスクの類似性で判断

事前学習データサイズ
のみでは不十分

③総当たりファインチューニングによる選択

- すべてのモデルを実際にファインチ

効果的だが、
計算コストが
モデル数に応じて増大

➤ **実用上の課題あり**

目的・アプローチ

予算（計算コスト）内で、大量のPCMから最適なモデルを効率的に選択できるようにする

- タスクごとにモデルを評価できる，効率的な方法を提案

2種類の効率的なモデル選択手法を提案

- プロキシベース
- 分布ベース

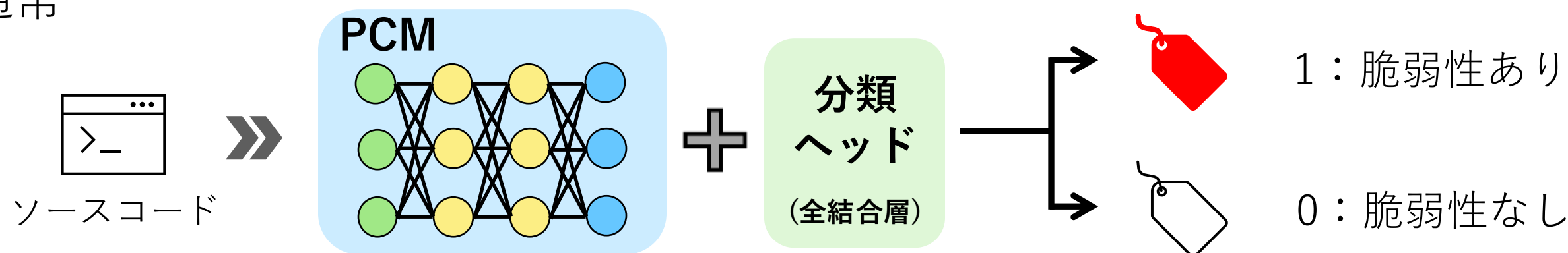
本発表では，脆弱性検出タスクを想定

- 入力：ソースコード，出力：1 or 0（脆弱性ありorナシ）

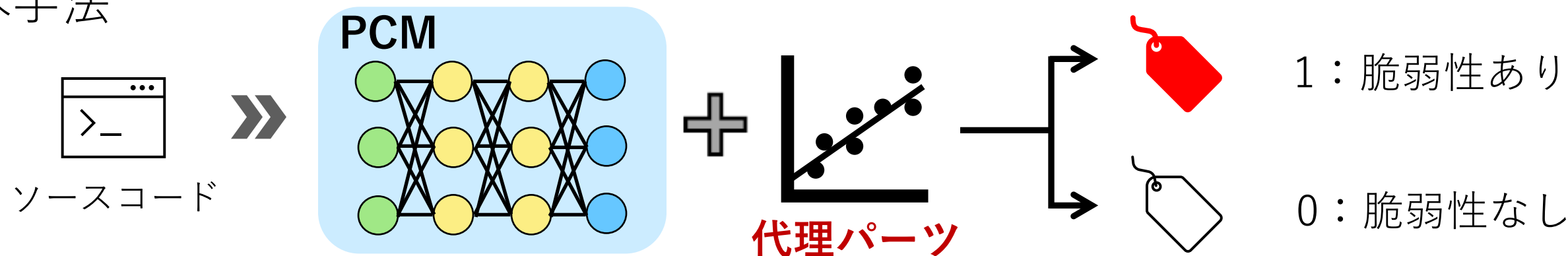
手法① | プロキシ（代理）ベース手法

軽量な代理パーツを用意・学習することで、簡易的な評価を行う

通常



本手法

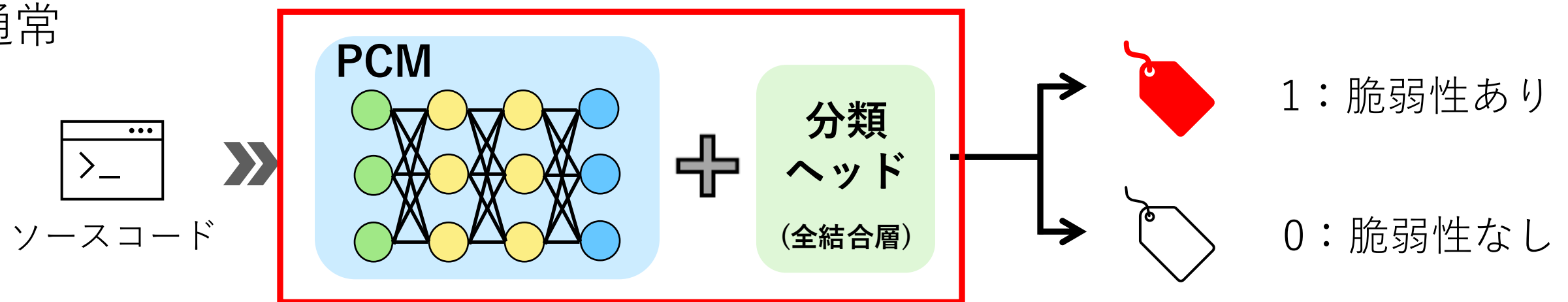


代理軽量パーツ&パラメータ固定で計算コストDOWN

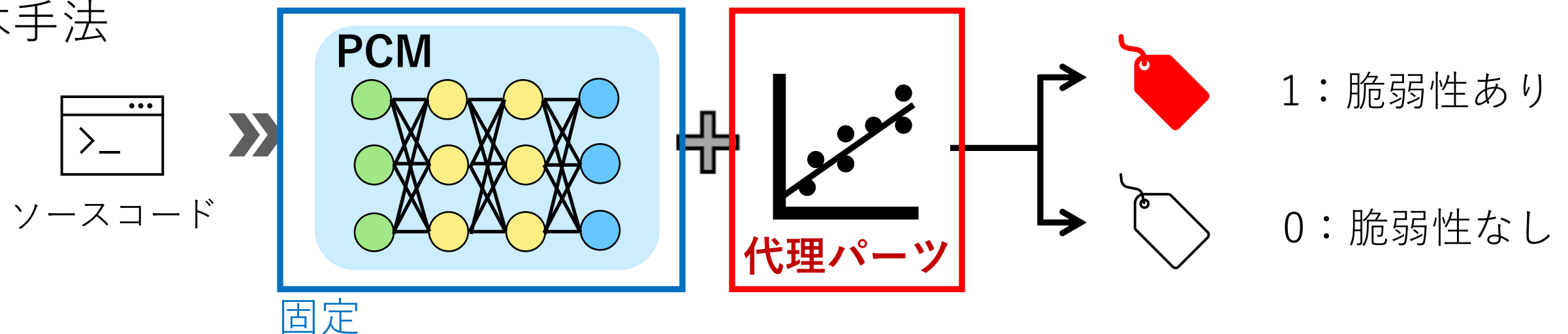
手法① | プロキシ（代理）ベース手法

軽量な代理パーツを用意・学習することで、簡易的な評価を行う

通常



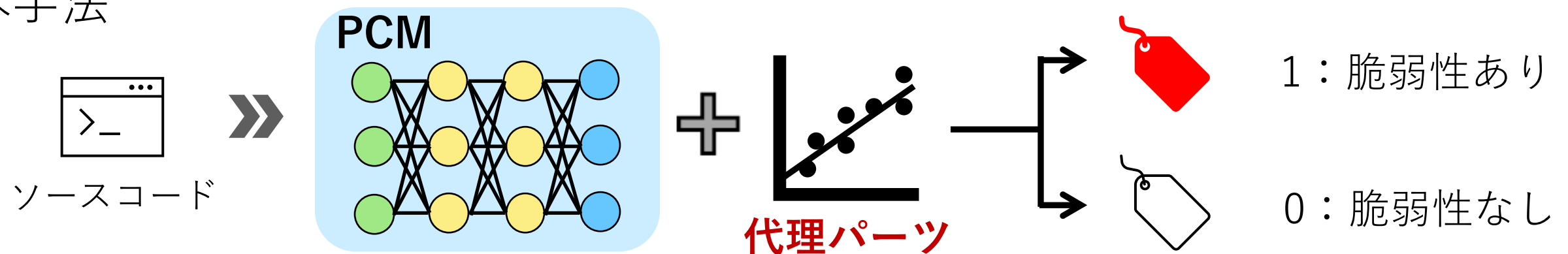
本手法



代理軽量パーツ&パラメータ固定で計算コストDOWN

手法① | プロキシ（代理）ベース手法

本手法



3種類のプロキシパーツを使用

- k-NN, 線形分類器, SVM

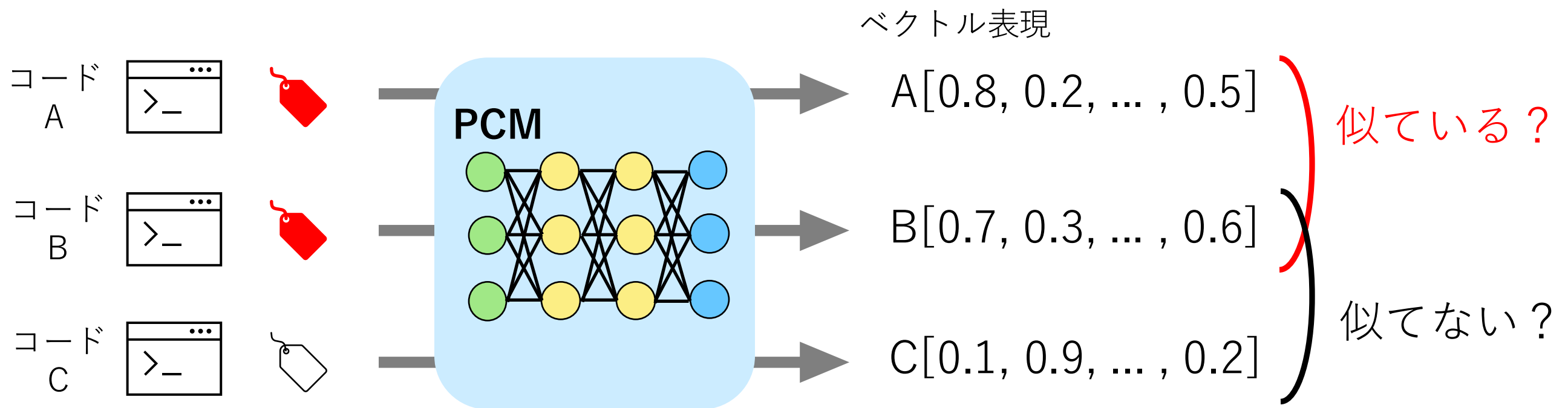
ポイント

- PCMをパラメータ固定のエンコーダとして利用
- 軽量な分類器を代理パーツとして使用, 特徴表現を学習
- 仮説: 良い特徴抽出さえできれば, 良い分類性能が出せる

手法② | 分布ベース手法

コードの特徴空間とラベルの特徴空間の類似度を計測することで評価を行う

- 「似ているコードは、同じラベルを持つべき」という仮説に基づくモデル評価・選択方法

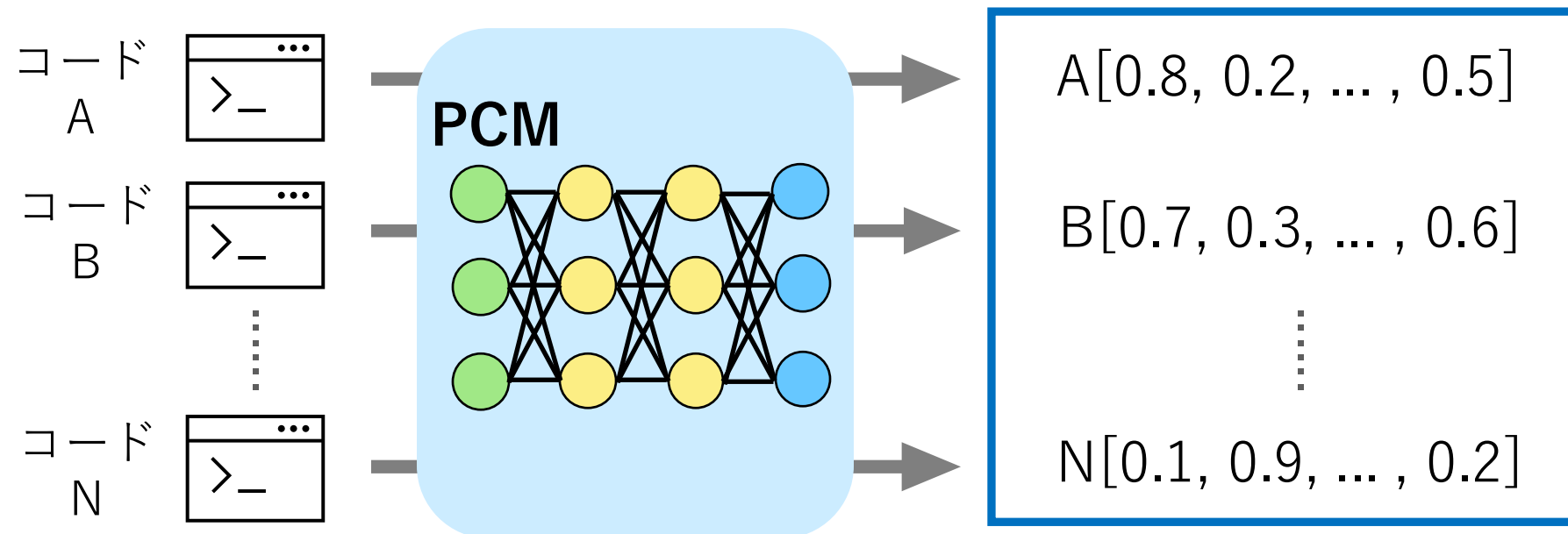


学習を行わず，特徴量のみの利用で計算コストDOWN

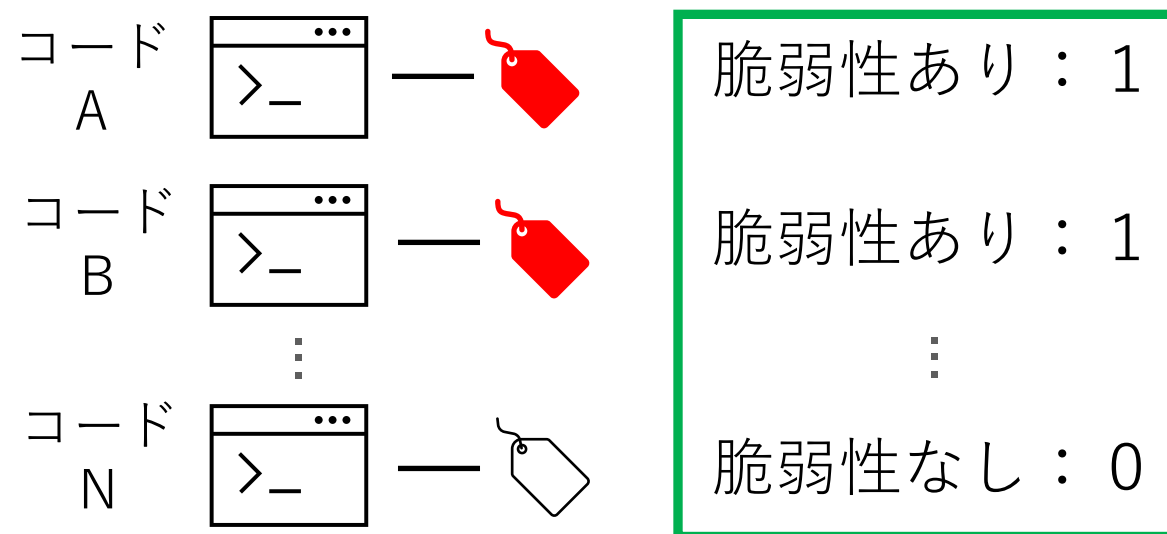
- プロキシベース同様，PCMは特徴量抽出のみに利用

手法② | 分布ベース手法の実行手順

① 一定数のサンプルコードから特徴量を抽出



② ラベルを1 or 0に置き換え



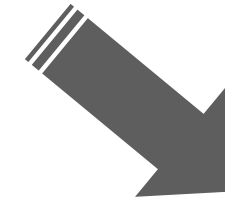
手法② | 分布ベース手法の実行手順

③コード同士の類似度行列を作成

A[0.8, 0.2, ..., 0.5]
B[0.7, 0.3, ..., 0.6]
⋮
N[0.1, 0.9, ..., 0.2]



	A	B	...	N
A	1	0.97	...	0.32
B	0.97	1	...	0.28
⋮	⋮	⋮	⋱	⋮
N	0.32	0.28	...	1



⑤行列間の類似度を算出

相関スコア

||
0.92



高いスコア

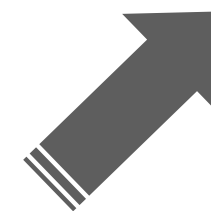
||
良いPCM

④ラベル同士の類似度行列を作成

A 脆弱性あり : 1
B 脆弱性あり : 1
⋮
N 脆弱性なし : 0



	A	B	...	N
A	1	1	...	0
B	1	1	...	0
⋮	⋮	⋮	⋱	⋮
N	0	0	...	1



手法② | 分布ベース手法の実行手順

Point

- コードの特徴行列と
ラベルの特徴行列が似ている

= 似たコードは同じラベルを持ち、
似てないコードは異なるラベルを持つ

→ **適切な特徴抽出**が行えている
(= そのPCMはタスクに適している)

実験設定①

3つのコード関連タスクで評価

- 脆弱性検出（27,318関数，二値分類）
- アルゴリズム分類（52,000プログラム，104クラス）
- プログラミング言語識別（237,803コード片，21言語）

100種類のオープンソースモデル

- CodeBERT系，PLBART系，CodeT5系，StarCoder系など
- モデルサイズ，事前学習データなどを区別して100種類
- Hugging Face，TensorFlow Hubなどオンラインリポジトリから収集

実験設定②

比較方法

- 提案手法と直感的手法①②(モデルサイズ, 学習データ)の比較
 - 従来の選択手法に対する優位性
- 提案手法と直感的手法③(総当たりFT)の比較
 - 総当たりFT手法をゴールドスタンダードとした, 提案手法との性能差

評価指標

- NDCG@k
 - 選択されたトップkモデルが, 理想的なトップkモデルとどれだけ一致するか
- Rel@k
 - 選択されたトップkモデル中の最良モデルと、全体での最良モデルの性能比

結果① | 性能差について

選択手法	脆弱性検出		アルゴリズム分類		プログラム言語識別	
	NDCG@5	Rel@5	NDCG@5	Rel@5	NDCG@5	Rel@5
モデルサイズ	0.49	0.44	0.77	0.97	0.49	0.58
データサイズ	0.34	0.34	0.69	0.98	0.50	0.60
プロキシベース	0.54	0.70	0.78	0.98	0.75	0.93
分布ベース	0.56	0.66	0.86	1.00	0.87	0.92
総当たりFT	1.00	1.00	1.00	1.00	1.00	1.00

提案手法は直感的な手法より一貫して高い性能

- アルゴリズム分類, プログラム言語識別で特に高精度
- Rel@10であれば, 脆弱性検出でも0.94と高い精度を実現

結果② | 効率差について

選択手法	選択時間	サンプル数
プロキシベース(k-NN)	95秒	1,000
分布ベース(H-Score)	210秒	1,000
モデル/データサイズ	<1秒	-
総当たりFT	約2,700時間	全データ

総当たりFT手法に比べ，選択時間を劇的に削減

- 100モデルの総当たりには約113日が必要
- プロキシベースは速度に優れ，分布ベースは精度に優れる
- サンプルを増やせば精度も向上するが，時間も増える

議論

直感的手法の限界

- より大きいモデル/データが常に良いわけではない
- 実際、より小さいモデルの方が性能が良かったケースも

タスクによる精度変化

- アルゴリズム分類では、特に分布ベースが効果的
- 一方、脆弱性検出では性能がやや低下

限界と課題

- 本研究は分類タスクのみが対象
- 生成タスクへの応用は今後の課題

まとめ

| PCMの効率的選択に向けた手法の提案

| 2種類の提案手法により，高精度・高効率で適切なモデル選択が可能

| 生成タスクへの拡張や，動的選択の実現が今後の課題

所感

やはり、応用性・実用性に欠ける

- 共同研究では、生成タスクを含む動的な選択が目標
- 提案手法をそのまま転用するのは難しそう

一方、計算の効率化・軽量化という考え方自体は興味深かった

- 共同研究の方でも、モデル評価における計算時間・コストが最大のボトルネックの一つ
- そのままの転用は難しいが、考え方としては参考になるかもしれない