Быстрое сравнение по образцу и обучение в глубину с помощью диаграмм решений

студент 545 группы Зубаревич Дмитрий

Научный руководитель: к.ф.-м.н., доцент кафедры информатики Бугайченко Д.Ю.

Рецензент: аспирант Дзюба А.А.

Цель работы

Разработка алгоритмов классификации на основе бинарных диаграмм решений.

Задачи:

- Разработать и реализовать требующиеся алгоритмы
- Создать систему тестирования для них
- Исследовать эффективность различных типов BDD
- Оценить качество результатов на распознавании рукописных символов базы MNIST

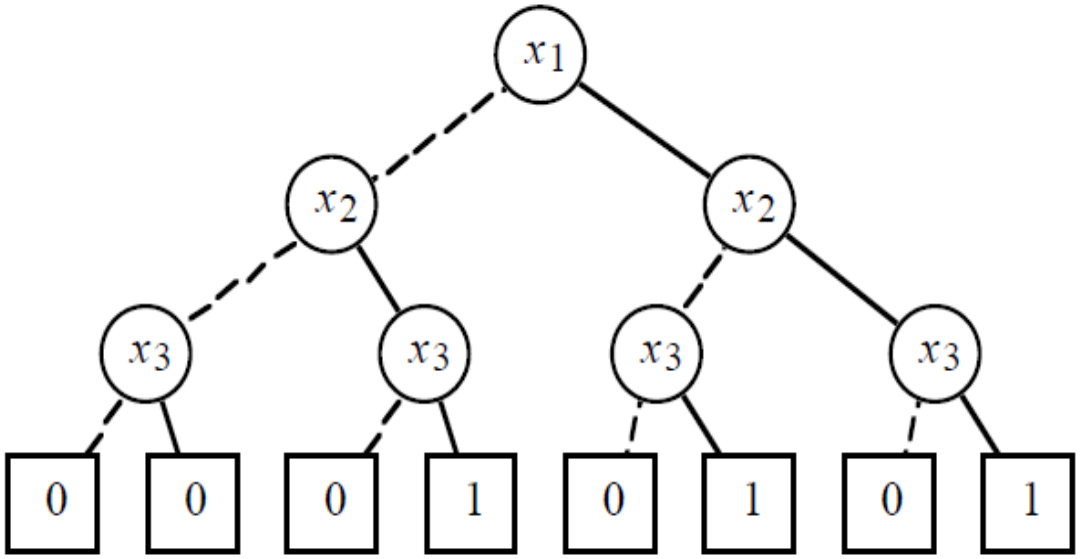
Бинарные диаграммы решений

- Кодирование функций графовой структурой
- Медленное построение, но быстрое использование
- Строятся путем редукции дерева принятия решений
- Интересные типы: классические BDD и ZDD

Пример

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

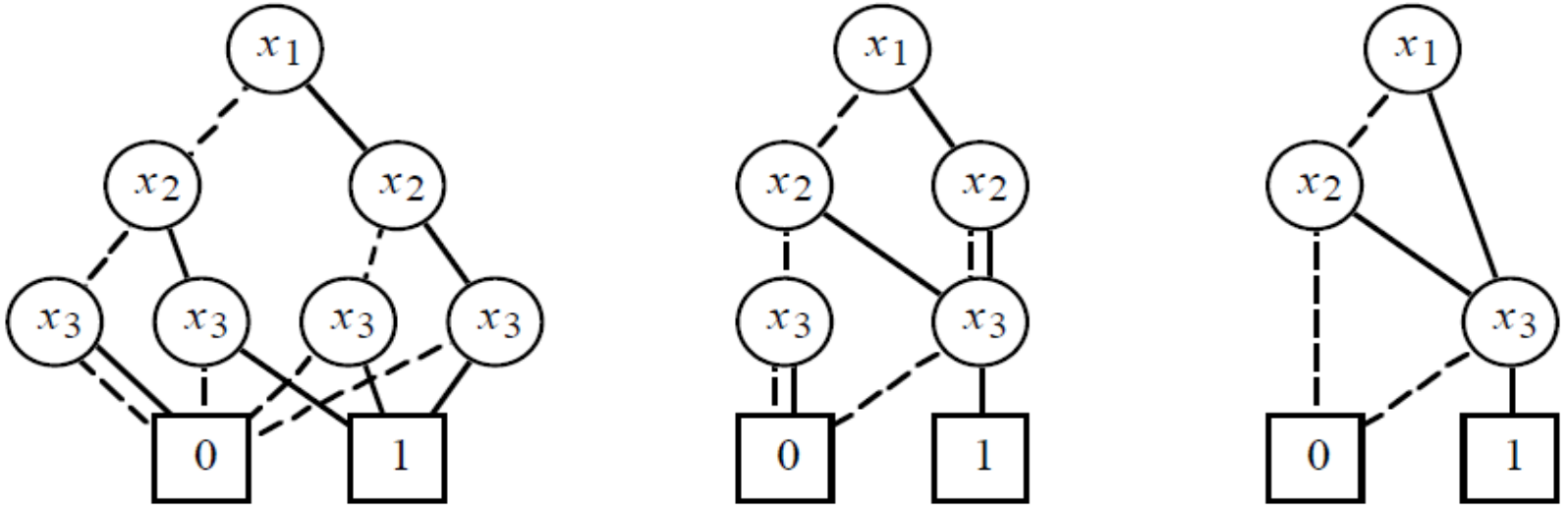
Таблица истинности



Дерево принятия решений

- f *- функция*
- x_1, x_2, x_3 - логические переменные
- 0, 1 – значения функции
- У каждого нетерминала два потомка: положительный и отрицательный

Классические BDD

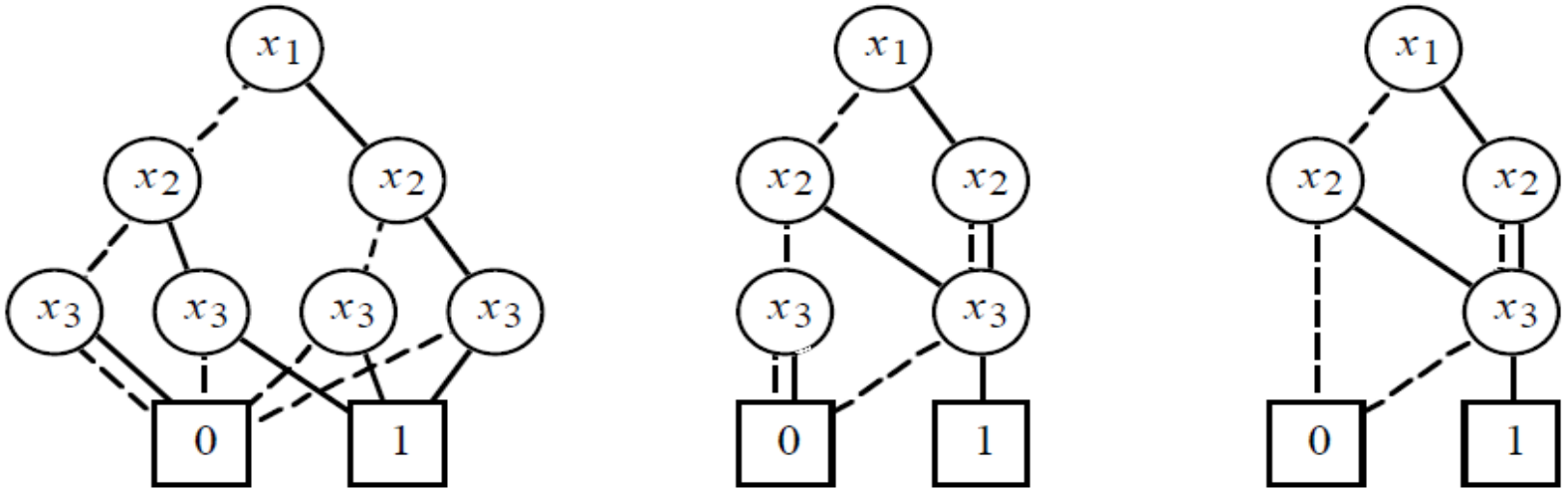


Применение правил редукции к дереву принятия решений

Правила редукции:

- ***Слияние*** изоморфных подграфов
- ***Удаление*** вершин, с одинаковыми потомками

ZDD



Применение правил редукции к дереву принятия решений

Правила редукции:

- ***Слияние*** изоморфных подграфов
- ***Удаление*** вершин, положительные дуги которых ведут в ноль терминал

Нейрон на основе BDD (ZDD)

Нейрон
BDD/ZDD 1 – хранение множества
BDD/ZDD 2 – функция схожести с образцами

Обучение:

- Образцы помещаем в множество
- Строим функцию схожести с множеством образцов

Реакция нейрона на объект:

- 0, если объект присутствует в множестве, иначе:
- Расстояние от объекта до множества образцов

Расстояние до множества

- S – множество всех возможных объектов
- $P \subset$ S – обучающее множество
- $sim: S \times S \to R$ – метрика
- $agg: R \times R \to R$ – функция агрегации
- $f_{P,sim,agg}: S \to R$ – функция сходства с множеством образцов:

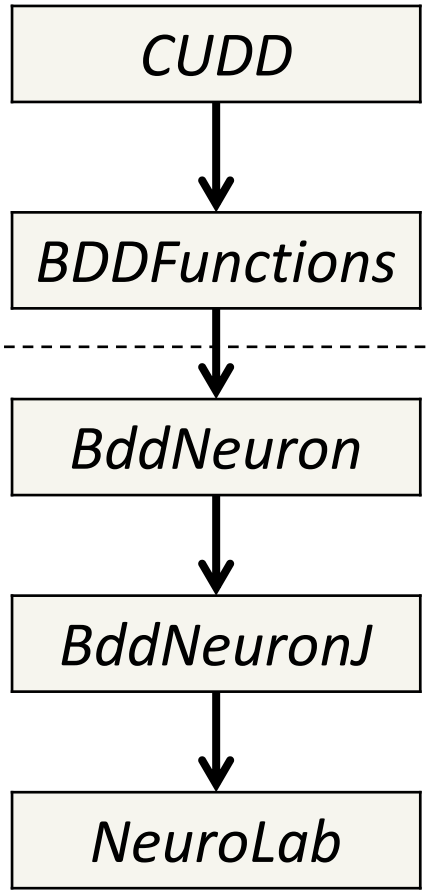
$$f_{P,sim,agg}(s) =$$

$$= \begin{cases} sim(s_0, s), P = \{s_0\} \\ agg\left(sim(s_0, s), f_{\{s_1,\dots,s_n\},sim,agg}(s)\right), P = \{s_0, \dots, s_n\} \end{cases}$$

Обучение в глубину

- Простейшая схема классификатора
  - Один класс – один нейрон
  - Поиск минимума среди расстояний
- Разрешение конфликтов
  - Более тонкое обучение пары нейронов
  - Последовательное добавление к простейшей схеме “разрешателей” конфликтов
- Комбинированная схема
  - Использование ответов нейронов для классических алгоритмов классификации (ANN, SVM)

Архитектура



Пакет решающих диаграмм – Colorado University Decision Diagram Package

Библиотека, предоставляющая гибкий объектно-ориентированный C++ интерфейс

Библиотека, реализующая нейрон, основанный на решающих диаграммах

Java-обертка для библиотеки *BddNeuron*

Библиотека для создания и тестирования классификаторов, использующих BDD/ZDD

Результаты тестирования

Классификатор	Точность	Время обучения
Простейшая схема:		
Среднее	60,66%	~1,5 часа
Минимум	91,31%	~12 часов
Комбинирование:		
Perceptron	92,57%	~12,5 часов
SVM	93,4%	~12,5 часов
Эталон:		
Perceptron	91,95%	~0,5 часа
SVM	92,34%	~0,5 часа

- 10 классов
- 6000 образцов на каждый класс
- 1000 тестовых примеров

Результаты тестирования

Классификатор	Точность	Время обучения
Разрешение конфликтов		
Среднее	93,64%	~6 часов
Комбинирование:		
Perceptron	94,37%	~6,25 часов
SVM	95,75%	~6,25 часов
Эталон:		
Perceptron	93,83%	~0,25 часа
SVM	94,25%	~0,25 часа

- 2 класса
- 6000 образцов на каждый класс
- 1000 тестовых примеров на каждый класс

Сравнение типов BDD

С точки зрения динамики роста количества узлов в диаграммах при обучении классификатора

- для функции схожести с множеством образцов лучше подходят классические BDD
- для множества образцов лучше подходят ZDD

Итоги

- Разработаны алгоритмы построения классификаторов на основе BDD, ZDD
- Создана система тестирования
- Проведены тесты на базе MNIST
- Проведено сравнение BDD и ZDD
- Поддержана работа с ZDD в BddFunctions
- Поддержана возможность сохранения/загрузки BDD в BddFunctions
- Реализована возможность распределенной работы алгоритмов
- По результатам работы принята статья на конференцию MLDM 2014