10. **Учене на група от модели**

• Комбиниране на множество модели
• Основната идея
• **Bagging**
• Разлагане на пристрастие и дисперсия, **bagging** с разходи
• Рандомизация
• **Random forests**, **rotation forests**
• **Boosting**
• **AdaBoost**, силата на **boosting**
• Адитивна регресия
• Числова прогноза, адитивна логистична регресия
• Интерпретируеми ансамбли
• **Option trees**, алтернативни дървета за решения, логистични моделни дървета
• **Stacking**

Комбиниране на множество модели

• Основна идея: изграждане на различни „експерти“, които гласуват.
• Предимство: Често подобрява предсказателната производителност.
• Недостатък: Обикновено произвежда изход, който е много труден за анализ.
• Но: има подходи, които целят да създадат една разбираема структура.

Bagging

• Комбиниране на прогнози чрез гласуване/усредняване.
• Всеки модел получава равно тегло.
• „Идеализирана“ версия:
• Вземане на няколко тренировъчни набора с размер n n n (вместо само един тренировъчен набор с размер n n n).
• Изграждане на класификатор за всеки тренировъчен набор.

• Комбиниране на прогнозите на класификаторите.
• Схемата за обучение е нестабилна, почти винаги подобрява производителността.
• Нестабилен обучаем: малка промяна в тренировъчните данни може да доведе до голяма промяна в модела (напр. при обучение на дървета за решения).

Bagging класификатори

Нека n е броят на екземплярите в тренировъчните данни.
За всяка от t итерации:
• Вземане на n екземпляра от тренировъчния набор (с връщане).
• Прилагане на алгоритъма за обучение върху извадката.
• Запазване на получения модел.
За всеки от t модела:
• Предсказване на класа на екземпляра с помощта на модела.
• Връщане на класа, който е предсказан най-често.

Рандомизация и random forests

• Може да се рандомизира алгоритъмът за обучение вместо входа.
• Някои алгоритми вече имат случаен компонент: напр. начални тегла в невронна мрежа.
• Повечето алгоритми могат да бъдат рандомизирани, напр. алчни алгоритми:
• Избиране на N опции на случаен принцип от пълния набор от опции, след което избор на най-добрата от тези N опции.
• Напр.: избор на атрибути в дървета за решения.
• По-общо приложим от **bagging**: напр. можем да използваме случайни подмножества в класификатор за най-близки съседи.
• **Bagging** не работи със стабилни класификатори като тези за най-близки съседи.
• Може да се комбинира с **bagging**.

• Когато се използва с дървета за решения, това води до известния метод **random forest** за изграждане на ансамблови класификатори.

Boosting

• **Bagging** лесно може да се паралелизира, защото членовете на ансамбъла се създават независимо.
• **Boosting** е алтернативен подход.
• Също използва гласуване/усредняване.
• Но: тегли моделите според производителността.
• Итеративен: новите модели се влияят от производителността на предишно изградените.
• Насърчава новия модел да стане „експерт“ за екземпляри, погрешно класифицирани от по-ранни модели.
• Интуитивно оправдание: моделите трябва да са експерти, които се допълват взаимно.
• Съществуват много варианти на **boosting**, ние разглеждаме няколко.

Boosting с AdaBoost.M1

Присвояване на равно тегло на всеки тренировъчен екземпляр.
За t t t итерации:
• Прилагане на алгоритъма за обучение върху претегления набор от данни, запазване на получения модел.
• Изчисляване на грешката е е е на модела върху претегления набор от данни.
• Ако e=0 или e≥0.5:
Прекратяване на генерирането на модели.
• За всеки екземпляр в набора от данни:
Ако е класифициран правилно от модела:
Умножаване на теглото на екземпляра по e/(1−e)

• Нормализиране на теглото на всички екземпляри.
Класификация:
• Присвояване на тегло = 0 на всички класове.
• За всеки от t t t (или по-малко) модела:
За класа, предсказан от този модел, добавяне на −loge/(1−e) към теглото на този клас.
• Връщане на класа с най-високо тегло.

Коментари за AdaBoost.M1

• **Boosting** изисква тегла … но
• Може да се адаптира алгоритъмът за обучение … или
• Може да се приложи **boosting** без тегла:
• Пресъздаване на данни с вероятност, определена от теглата.
• Недостатък: не всички екземпляри се използват.
• Предимство: ако грешката е > 0.5, може да се пресъздаде отново.
• Алгоритъмът за **boosting AdaBoost.M1** произлиза от работа в теорията на изчислителното обучение.
• Теоретичен резултат:
• Грешката при обучение намалява експоненциално с извършването на итерации.
• Други теоретични резултати:
• Работи добре, ако базовите класификатори не са твърде сложни и
• Грешката им не става твърде голяма твърде бързо с увеличаване на итерациите.

Stacking

• Въпрос: как да се изгради хетерогенен ансамбъл, състоящ се от различни типове модели (напр. дърво за решения и невронна мрежа)?
• Проблем: моделите могат да се различават значително по точност.
• Идея: за комбиниране на прогнозите на базовите обучаеми, вместо

просто да се гласува, да се използва мета-обучаем (**meta learner**).
• В **stacking**, базовите обучаеми се наричат още модели от ниво-0 (**level-0 models**).
• Мета-обучаемият се нарича модел от ниво-1 (**level-1 model**).
• Прогнозите на базовите обучаеми са вход за мета-обучаемия.
• Базовите обучаеми обикновено са различни схеми за обучение.
• Предупреждение: не може да се използват прогнози върху тренировъчните данни за генериране на данни за модела от ниво-1!
• Вместо това се използва схема, базирана на кръстосана валидация (**cross-validation**).