7. **Разширяване на instance-based learning и линейни модели**

Instance-based learning

- Подрязване и намаляване на броя на примерите
- Претеглени атрибути
- Обобщени примери и функции за разстояние
- Разширяване на линейни модели

• **Support vector machines**, **kernel ridge regression**, **kernel perceptrons**

- Multilayer perceptrons и мрежи с радиални базисни функции
- Gradient descent

• Числова прогноза с локални линейни модели

- **Model Trees**
- Извличане на набори от правила с **model trees**
- Локално претеглена линейна регресия

Instance Based Learning - (обучение на база на примери) е метод за машинно обучение, при който моделът не конструира явна генерализирана функция по време на обучението. Вместо това, той съхранява обучителните примери (инстанции) и използва тяхната прилика с нови входни данни, за да направи прогнози. Един от най-известните методи в тази категория е k-nearest neighbors (k-NN).

Практически проблеми на 1-nearest-neighbour (1-NN) схемата и решения:

• Бавен (но: съществуват бързи подходи, базирани на дървета) - Решение: премахване на нерелевантни данни

• Шум (но: **k-NN** се справя доста добре с шума)-Решение: премахване на шумни инстанции

• Всички атрибути се считат за еднакво важни- Решение: претегляне на атрибути (или просто избор)

• Не извършва явна обобщаване-Решение: подход на **rule-based NN**

Learning prototypes- Обучение на прототипи - При него целта е да се намали броят на съхраняваните примери, като се запазят само най-представителните (прототипни) инстанции. Това подобрява ефективността на **instance-based learning**, като намалява нуждата от памет и ускорява класификацията, без да се жертва точността.

Основни принципи:

-Запазване само на съществените инстанции - Само тези примери, които реално влияят върху вземането на решение, трябва да се съхраняват. Излишните инстанции, които не допринасят към класификацията, могат да бъдат премахнати.

-Филтриране на шумните инстанции - Шумът в данните (грешни или нерепрезентативни примери) може да доведе до грешни прогнози. За да се подобри точността, такива инстанции трябва да бъдат премахнати или коригирани.

-Използване на само прототипни примери - Вместо да се съхраняват всички обучителни данни, може да се запазят само най-представителните инстанции. Тези "прототипи" са обобщени версии на множеството примери и позволяват по-бърза и ефективна класификация.

Предимства на обучението с прототипи: Намалява необходимата памет; Ускорява процеса на класификация; Намалява чувствителността към шум в данните; Подобрява интерпретируемостта на модела

Тази техника се използва в методи като **Condensed Nearest Neighbors (CNN)** и **Reduced Nearest Neighbors (RNN)**, които избират подмножество от най-полезните примери за класификация.

Speed up classification, combat noise (Ускоряване на класификацията, справяне с шума), David Aha's IB2

- Основна идея: **пестене на памет** и **ускоряване на класификацията**
- **Работа поетапно (incrementally)** – алгоритъмът се подобрява с всяка нова инстанция
- **Добавяне само на погрешно класифицирани примери**
- **Проблем**: Шумни данни също се включват, което може да влоши точността

David Aha's IB3 – подобрение за работа с шум

- Изхвърляне на слабо представящи се примери
- Изчисляване на доверителни интервали за:
  1. Успеваемост на всяка инстанция
  2. Базова точност за класа на инстанцията
- Приемане/отхвърляне на примери въз основа на представянето им:
  1. Приемане**,** ако долната граница на (1) надвишава горната граница на (2)
  2. Отхвърляне, ако горната граница на (1) е под долната граница на (2)

Weight attributes (Претегляне на атрибути), David Aha's IB4

- Претегляне на всеки атрибут (**weights can be class-specific**)
- **Претеглено Евклидово разстояние:** едно от най-често използваните мерки за разстояние в метричните пространства. То се дефинира като дължината на отсечката между две точки в многомерното пространство. В двумерното пространство, например, Евклидовото разстояние между две точки $A(x_1,y_1)$ и $B(x_2,y_2)$ се изчислява по формулата:

$$d(A,B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

При **претегленото Евклидово разстояние** се въвеждат тегла за различните координати (или измерения) на пространството. Това се прави, за да се отчете различната важност на всяко измерение.

Формулата за **претеглено Евклидово разстояние** между две точки A(x1,x2,...,xn) и B(y1,y2,...,yn) в **n-мерното пространство** е:

$$d(A,B) = \sqrt{w_1(x_1 - y_1)^2 + w_2(x_2 - y_2)^2 + ... + w_n(x_n - y_n)^2}$$

където:

- w1,w2,...wn са теглата, които определят значимостта на всяка координата.
- Ако всички тегла са равни на **1**, тогава получаваме стандартното Евклидово разстояние.

Защо се използва претеглено Евклидово разстояние?

- **Отчитане на различната важност на измеренията** – в много реални приложения някои променливи са по-важни от други.
- **Подобряване на алгоритми за машинно обучение** – например при **k-NN (k-близки съседи)** и **клъстеризация**.
- **Намаляване на влиянието на измерения с различни мащаби** – ако някои координати имат по-големи стойности от други, те ще доминират разстоянието. Използването на тегла може да балансира този ефект.

- **Актуализиране на теглата въз основа на най-близкия съсед:**
  - **Ако класът е правилен** → увеличаване на теглото
  - **Ако класът е грешен** → намаляване на теглото
  - Количеството на промяна за даден атрибут зависи от разликата $|x_i - y_i|$
  - Generalized exemplars (Обобщени примери)

- Обобщаване на инстанции в **хиперправоъгълници**

- **Онлайн**: динамично модифициране на правоъгълниците
- **Офлайн**: намиране на минимален брой правоъгълници, които покриват всички инстанции

Ключови дизайнерски решения:

- **Позволяване на припокриващи се правоъгълници? →** Изисква стратегия за разрешаване на конфликти
- **Позволяване на вложени (nested) правоъгълници?**
- **Как да се справим с непокрити инстанции?**

Rectangular generalizations (Правоъгълни обобщения)

- **Правилото за най-близък съсед (nearest-neighbor rule) се прилага извън правоъгълниците**
- **Правоъгълниците са правила!** (но могат да бъдат по-консервативни от стандартните правила)
- **Вложените правоъгълници представляват правила с изключения**

Separating generalized exemplars - Разделяне на обобщените примери чрез по-гъвкави стратегии

Generalized distance functions (Обобщени функции за разстояние)

Проблем: **Евклидовото разстояние** работи добре само за чисто **числови** данни. **Решение**: Подход, базиран на трансформации на атрибутите

- **K*** *similarity* → вероятността дадена инстанция **A** да се трансформира в **B** случайно
  - Усредняване по всички възможни трансформационни пътища
  - Претегляне на пътищата според вероятността им

- **Методът осигурява:** Унифициран подход за работа с различни типове атрибути; Лесно разширяване към изчисляване на разстояния между групи от инстанции

Discussion and Bibliographic Notes (Дискусия и библиографски бележки)

- **Методите на най-близкия съсед (nearest-neighbor)** добиват популярност благодарение на **Aha (1992)**
- **Salzberg (1991)** предлага, че генерализацията чрез вложени примери може да постигне висока точност
- **Wettschereck и Dietterich (1994)** оспорват това и показват, че резултатите не се пренасят добре в други домейни
- **Martin (1995)** изследва проблема със свръхгенерализацията при припокриващи се хиперправоъгълници
- **Обобщената функция за разстояние, базирана на трансформации, е предложена от Cleary и Trigg (1995)**

Extending Linear Models (Разширяване на линейните модели)

Support Vector Machines (SVM) - алгоритми за обучение на линейни класификатори

- **Предимства:**
  ✓ Устойчиви на **overfitting**
  ✓ Намерят **оптимална линейна разделителна хиперплоскост (maximum margin hyperplane)**
  ✓ Бързи дори при нелинейни случаи
  ✓ Използват **математически трик** (kernel trick), за да избегнат нуждата от създаване на изкуствени атрибути
  ✓ Нелинейното пространство се създава **имплицитно**

Този подход обединява мощни техники за работа с **instance-based learning**, като включва оптимизации за **скорост**, **намаляване на**

шума, **претегляне на атрибути** и **разширяване на линейните модели** чрез **Support Vector Machines**.

The maximum margin hyperplane – хиперплоскост с максимален марж - примерите, които са най-близо до хиперплоскостта с максимален марж, се наричат support vectors. Support vectors - определят **maximum margin hyperplane**.

Finding support vectors

- **Support vector**: обучителен пример, за който **αi > 0**. Определяне на **αi** и **b**? - квадратичен проблем с ограничения. Съществуват стандартни инструменти за решаване, но специализирани алгоритми са по-бързи. Пример: **Platt's sequential minimal optimization (SMO) algorithm**. Досега методът предполага **separable data** (разделими данни).Nonlinear SVMs - Можем да създадем nonlinear classifier, като генерираме нови "pseudo" attributes от оригиналните атрибути. „Pseudo" атрибутите представляват комбинации от оригиналните атрибути. Пример: всички **polynomials of degree 2** (полиноми от втора степен), които могат да се формират от първоначалните атрибути. **Linear SVM** в разширеното пространство = **non-linear classifier** в оригиналното пространство. **Overfitting** не е голям проблем, защото **maximum margin hyperplane** е стабилна. В повечето случаи броят на **support vectors** е значително по-малък от броя на обучителните примери. Основен проблем: **Computation time** (изчислителна сложност). Всеки път, когато изчисляваме **dot product**, трябва да включим всички „pseudo attributes".

A mathematical trick - Избягване на директно изчисляване на „pseudo attributes“. Изчисляване на **dot product** преди извършване на нелинейното преобразуване. Това съответства на вграждане в **instance space**, който е представен от всички **products of n attributes**.

Other kernel functions - Трансформацията се нарича **kernel function**.

Noise

- Досега предполагаме, че данните са **separable** (в оригиналното или трансформираното пространство).
- **SVMs** могат да се използват за **noisy data**, като се въвежда **“noise” parameter C** (регулиращ параметър).
- **C** ограничава влиянието на който и да е обучителен пример върху **decision boundary**.
- Ограничението: **$0 \leq \alpha i \leq C$**.
- Това позволява да се максимизира **soft margin** (мек марж).
- Все още остава **quadratic optimization problem**.
- **C** трябва да се избере експериментално.

Sparse data - **SVM algorithms** са много по-бързи, ако данните са **sparse** (има много нулеви стойности). Защо? Защото алгоритъмът изчислява много **dot products**. При **sparse data** може да се пресмятат **dot products** много ефективно. Обработват **sparse datasets** с **10,000+ attributes**.

Support vector regression

- **Maximum margin hyperplane** важи само за **classification**.
- Идеята за **support vectors** и **kernel functions** може да се използва и за **regression**.
- Основният метод е подобен на **linear regression**, но:
    - **Разлика А:** Игнорира се грешка, по-малка от **ε**, и се използва **absolute error** вместо **squared error**.

    - **Разлика B:** Едновременно се стремим да максимизираме **flatness** на функцията.
    - Потребителят задава **ε**, който дефинира „тръба“ (**tube**).

The kernel perceptron

- Използва **kernel trick**, за да направи **non-linear classifier** на база **perceptron learning rule**.
- **Наблюдение:** При **perceptron learning rule**, тегловият вектор се модифицира чрез добавяне или изваждане на обучителни примери.
- Може да представим тегловия вектор чрез **всички неправилно класифицирани примери**.
- Това позволява използването на **kernel trick**.

Multilayer perceptrons (MLP) - **Използването на** kernels **е само един начин за изграждане на** non-linear classifier **чрез** perceptrons**.** MLP **е вид** artificial neural network**. Състои се от:** Input layer**,** Hidden layer(s)**,** Output layer**.** MLP структурата **обикновено се намира чрез експерименти. Параметрите се обучават чрез** backpropagation**.**

Backpropagation - Как да намерим теглата за дадена **MLP структура**? **Не може** да се използва **perceptron learning rule**, защото има **hidden layers**. **Минимизираме** грешката чрез **gradient descent**. **Loss function** трябва да осигурява градиентна информация. **Използваме sigmoid function вместо threshold function**. **Loss function**: **squared error**. Пример: $\mathbf{E = (y - f(x))^2}$.

Gradient descent example: Функция: $\mathbf{x^2 + 1}$**;** Производна: **2x; Learning rate**: **0.1;** Начална стойност: **4;** Намереният минимум може да е **локален**.

Avoiding overfitting

- **Early stopping**: Проверка върху **validation set**.

- **Weight decay**: Добавяне на наказателен термин към **error function**.

Speeding up learning

- **Momentum**: Повторно използване на пропорция от предишната промяна в теглата.
- **Втори ред оптимизационни методи**.

Discussion and Bibliographic Notes

- **SVMs** произлизат от **statistical learning theory** (**Vapnik, 1999**).
- **Soft-margin SVMs – Cortes & Vapnik (1995)**.
- **SMO algorithm – Platt (1998)**.
- **Kernel perceptron – Freund & Schapire (1999)**.
- **Support vector regression – Schölkopf et al. (1999)**.
- **Gradient methods for SVMs – Kivinen et al. (2002), Zhang (2004), Shalev-Shwartz et al. (2007)**.