Félig-felügyelt tanulás kernelekkel és hashing módszerek

Bodó Zalán

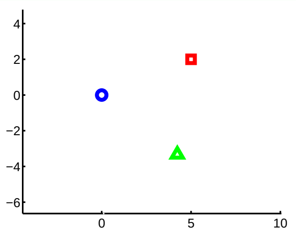
Babeş–Bolyai Tudományegyetem, Kolozsvár
Matematika és Informatika Kar
Magyar Matematika és Informatika Intézet

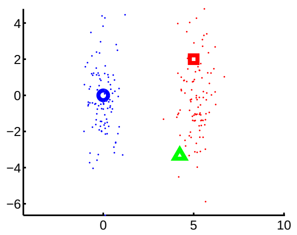
Tartalom

Félig-felügyelt tanulás kernelek segítségével

Felügyelt és félig-felügyelt tanulás

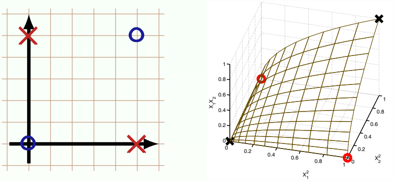
- felügyelt:
  $D = \{(\mathbf{x}_i, y_i) \,|\, \mathbf{x}_i \in X \subseteq \mathbb{R}^d, y_i \in \{-1, +1\}, i = 1, \ldots, \ell\}$;
  keressük $f : X \to \{-1, +1\}$-et, amely egyezik D-vel
- félig-felügyelt:
  $D = \{(\mathbf{x}_i, y_i) \,|\, i = 1, \ldots, \ell\} \cup \{\mathbf{x}_j \,|\, j = \ell + 1, \ldots, \ell + u =: N\}$, $\ell \ll u$;
  - **induktív**: keressük $f : X \to \{-1, +1\}$-et, amely egyezik D-vel + D_U felhasználása
  - **transzduktív**: keressük $f : D_U \to \{-1, +1\}$-et felhasználva a $D = D_L \cup D_U$ adatokat





Kernel módszerek a gépi tanulásban

- James Mercer (1909): minden folytonos szimmetrikus pozitív definit kernel függvény felírható skalárszorzatként egy (nagydimenziós) térben
- kernel: $k(\boldsymbol{x}, \boldsymbol{z}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \rangle = \phi(\boldsymbol{x})' \phi(\boldsymbol{z})$, feature leképezés: $\phi : X \rightarrow \mathcal{H}$
- alkalmazható minden geometriai konstrukció esetén, ha az szögeket, hosszokat és távolságokat használ



(a) (b)

1. ábra. Az XOR feladat: (a) input térben, (b) másodfokú polinomiális kernel által generált térben.

Gyakran használt kernelek:

- lineáris:

$$k(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{x}'\boldsymbol{z}$$

- polynomiális:

$$k(\boldsymbol{x}, \boldsymbol{z}) = (a\boldsymbol{x}'\boldsymbol{z} + b)^c$$

- Gauss-féle, RBF:

$$k(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{z}\|^2}{2\sigma^2}\right)$$

- szigmoid:

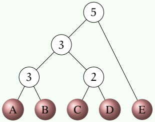
$$k(\boldsymbol{x}, \boldsymbol{z}) = \tanh(a\boldsymbol{x}'\boldsymbol{z} + r)$$

Adatfüggő (data-dependent) kernelek

- hagyományos kernelek: $D_1 \neq D_2$ adathalmazok esetén, $\boldsymbol{x}, \boldsymbol{z} \in D_1 \cap D_2$
$$k(\boldsymbol{x}, \boldsymbol{z}) = k(\boldsymbol{x}, \boldsymbol{z})$$
- adatfüggő kernelek: $D_1 \neq D_2$ adathalmazok esetén, $\boldsymbol{x}, \boldsymbol{z} \in D_1 \cap D_2$
$$k(\boldsymbol{x}, \boldsymbol{z}; D_1) \Bumpeq k(\boldsymbol{x}, \boldsymbol{z}; D_2)$$
ahol „$\Bumpeq$”-et „nem feltétlenül egyenlő”-ként olvassuk
- felügyelt tanulás + adatfüggő kernelek = **félig-felügyelt tanulás**

Klaszter-kernelek

- **klaszter feltevés** (*cluster assumption*): ha két pont ugyanabban a klaszterben van, nagy valószínűséggel ugyanabba az osztályba is tartoznak (= ua. a címkéjük)
- *hierarchikus klaszter-kernel*:
  - használjuk a hierarchikus klaszterezésből származó távolságokat (single, complete, average linkage)
  - hasznos tulajdonság: ultrametrikus távolságmátrixot eredményez az adatokon



Tétel [Fischer, 2003]

Adott ultrametrikus $\boldsymbol{M}$ esetén a $-\frac{1}{2}\boldsymbol{M}^c = -\frac{1}{2}\boldsymbol{JMJ}$ mátrix egy Gram-mátrix lesz, amely a $\boldsymbol{z}_i$, $i = 1, 2, \ldots, N$ vektorok skalárszorzatait tartalmazza, melyek euklideszi távolságai az $\boldsymbol{M}$-ben vannak ($\boldsymbol{J} = \boldsymbol{I} - (1/N)\mathbf{1}\mathbf{1}'$).

  - az algoritmus: (i) pontok hierarchikus klaszterezése, (ii) M_{ij} = linkage távolság az eredő ultrametikus fában a *legközelebbi közös felmenőnél*, $M_{ii} = 0$.

- *átsúlyozó klaszter-kernelek*:
  - kernel kombinációk: $\boldsymbol{K}_1 + \boldsymbol{K}_2$, $a \cdot \boldsymbol{K}$, $\boldsymbol{K}_1 \odot \boldsymbol{K}_2$
  - átsúlyozó klaszter-kernel: $\boldsymbol{K} = \boldsymbol{K}_{\mathrm{rw}} \odot \boldsymbol{K}_{\mathrm{b}}$, ahol
    - $\boldsymbol{K}_{\mathrm{b}}$ = alap kernel (pl. Gauss, polinomiális stb.)
    - $\boldsymbol{K}_{\mathrm{rw}}$ = átsúlyozó kernel
  - Gauss-féle átsúlyozó kernel:

$$k_{\mathrm{rw}}(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{U}_{\cdot\boldsymbol{x}} - \boldsymbol{U}_{\cdot\boldsymbol{z}}\|^2\right)$$

  - skalárszorzat alapú átsúlyozó kernelek:

$$\boldsymbol{K}_{\mathrm{rw}} = \boldsymbol{U}'\boldsymbol{U} + \alpha \cdot \mathbf{11}', \quad \alpha \in [0, 1)$$

$$\boldsymbol{K}_{\mathrm{rw}} = \beta \cdot \boldsymbol{U}'\boldsymbol{U} + \mathbf{11}', \quad \beta \in (0, \infty)$$

    ahol $\boldsymbol{U}$ a $K \times N$ méretű klaszter hovatartozási mátrix (oszlopokban)
- **probléma**: *általánosítás* – közelítő módszerek

Legközelebbi szomszédok gyors keresése

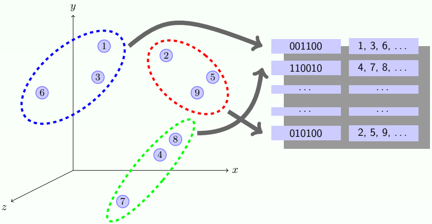
Legközelebbi szomszédok keresése pozitív definit kernelekkel

- egyenlőtlenségek:

$$d^2(\mathbf{x},\mathbf{z}) \geq k(\mathbf{x},\mathbf{x}) + k(\mathbf{z},\mathbf{z}) - 2\sqrt{k(\mathbf{x},\mathbf{x})k(\mathbf{z},\mathbf{z})}$$

$$d^2(\mathbf{x},\mathbf{z}) \geq \frac{1}{2}k(\mathbf{z},\mathbf{z}) - k(\mathbf{x},\mathbf{x})$$

- tároljuk a maximális d^2_{max} távolságot, és mindig megnézzük, hogy pl. $\frac{1}{2}k(\mathbf{z},\mathbf{z}) - k(\mathbf{x},\mathbf{x}) \geq d^2_{max}$
- a számítások akár 50%-át megtakaríthatjuk



2. ábra. Locality-sensitive hashing

Locality-sensitive hashing[1] véletlen hipersíkokkal

- Hamming-távolság hatékonyan kiszámítható:

$$d_{\mathrm{H}}(\boldsymbol{u}, \boldsymbol{v}) = \sum_{i=1}^{n} (\boldsymbol{u} \,\mathrm{XOR}\, \boldsymbol{v})$$

- a k darab hash függvényünk legyen

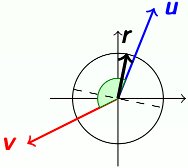
$$h_i(\boldsymbol{u}) = \begin{cases} 1, & \boldsymbol{r}_i'\boldsymbol{u} \geq 0 \\ 0, & \boldsymbol{r}_i'\boldsymbol{u} < 0 \end{cases}, i = 1, 2, \ldots, k, \boldsymbol{r} \in \mathbb{R}^d \text{ véletlen vektor}$$

- egy $\boldsymbol{x}$ ponthoz rendelt bináris hash szekvencia

$$[h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_k(\boldsymbol{x})]$$

- ekkor:

$$P(h_{\boldsymbol{r}}(u) = h_{\boldsymbol{r}}(v)) = 1 - \frac{\theta(\boldsymbol{u}, \boldsymbol{v})}{\pi}$$



[1] Helyérzékeny hasítás/hashelés

Hash kulcsok kiterjesztése címkézett adatok segítségével

- feltételezzük, hogy vannak címkézett pontjaink (minden osztályból)
- valamilyen hashing módszerrel előállított bináris kulcsok + hibajavító kódok segítségével tanított félig-felügyelt módszerek (ECOC)
- ECOC:
  - kódmátrix:

$$\boldsymbol{M} = \begin{bmatrix} 1 & 1 & 1 & -1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & -1 & 1 & 1 \end{bmatrix}$$

  - $\boldsymbol{M}$ (osztályok száma $\times c$) minden oszlopára betanítunk egy *félig-felügyelt* modellt
  - kimenet: $f_i(\boldsymbol{x})$ = a FF módszer kimenete, $i = 1, \ldots, c$
- kiterjesztett kulcsok:

$$[h_1(\boldsymbol{x}), \ldots, h_k(\boldsymbol{x}), \underbrace{f_1(\boldsymbol{x}), \ldots, f_c(\boldsymbol{x})}_{\text{kiterjesztés}}]$$

Köszönöm a figyelmet!