

1 Úvod

Bude se pracovat v Matlabu, v moodle je skupina.

Poznámka (Úkoly počítačového vidění)

Detekovat, najít a určit věci (tváře, jestli se smějí, nádory, znaky jako znaky na SPZ, biometrika jako oko, tvář, podpis, budovy, lidi, auta, ...) na obrázku

Poznámka (Rozpoznávání objektu tradičním náhledem)

pixels -> určení feature (např. rozdělení oblastí podle tvarů, důležitých bodů, barev, umístění hledaného objektu) expertem -> učení klasifikátoru -> rozpoznávání

Definice 1.1 (Feature vektor)

Univerzální převedení obrázku do důležitých věcí.

Měl by být invariantní (měl by být stejný při rotaci a škálování), diskriminační (dobře rozdělovat objekty), kompaktní (co nejmenší)

Definice 1.2 (Rozpoznávání)

Feature vektory tvoří prostor, kde se algoritmus naučí najít hranici, která odděluje objekty, co jsou nějaké a co jsou jinaké.

Poznámka (Klasifikace může být za pomoci)

Statistiky – Bayesova teorie rozhodování

Pravidel – Rozhodovací strom

Metriky – Technika nejbližšího souseda, diskriminační analýza?, podpůrné vektorové stroje?

Biologické inspirace – Neuronové sítě

Definice 1.3 (Učení s učitelem)

Na training setu víme správné odpovědi.

Definice 1.4 (Naivní Bayesův klasifikátor)

Vychází z podmíněné pravděpodobnosti na základě věcí, co víme.

┌
 Například

└ Rozeznávání falešného úsměvu. 91,3%.

Definice 1.5 (Rozhodovací stromy)

Pravidly určíme, kterou větví se vydáme. Výhodou je, že nepotřebujeme koncept vzdálenosti.

Například

Rozpoznávání, co se děje na videu (např. vražda). 70% - 100%.

Definice 1.6 (K nejbližších sousedů)

Podíváme se na nejbližší známé objekty a rozhodneme se podle nich.

Například

Čtení znaků. 99% čísla, 94% velká a 89% malá písmena.

Definice 1.7 (Lineární klasifikace)

Rozdělení prostoru nadrovinou. Zlepšením je tzv. podpůrné vektorové stroje? (support vector machines)

Například

Rozpoznávání lidí a věku. 66,9 - 80% lidí, 63,8 - 75,7% věk.

Definice 1.8 (Umělé neuronové sítě)

Sítě z neuronů, které jsou velmi jednoduše simulovány, viz moje maturitní práce. (Na GitHubu pod uživatelem JoHavel).

Například

Rozpoznávání tváře. 90%. (80% na portrétech.)

Poznámka (Hluboké učení)

pixels -> učení se včetně feature -> rozpoznávání

Například

AlexNet (top 5 error cca. 16%)

Každoročně se pořádá ILSVRC (Imagenet Large Scale Visual Recognition Challenge), kde už se dosáhlo méně než 4% chyby (152 vrstev NN)...

Poznámka (Kombinovaný přístup)

pixely \rightarrow featury nalezené NN \rightarrow trénování klasifikátoru \rightarrow rozpoznávání

Hluboké učení se ale zdá účinnější.

Definice 1.9 (Klasifikační pipeline)

Features \rightarrow Výběr featur, jejich normalizace, ... \rightarrow klasifikace (výběr klasifikátoru, trénování klasifikátoru a následná klasifikace) \rightarrow evaluace \rightarrow features (respektive výstup, pokud jsme spokojeni).

Definice 1.10 (Features)

Vlastnosti objektu, spojují nějakým způsobem podobné objekty. Musí být diskriminativní (pokud nejsou dostatečně diskriminativní, jakože často nejsou, dá se ještě hledat rozdělení s nejmenší chybou). Měly by být kompaktní (co nejmenší, protože s příliš featurami nelze ve stejném čase dostatečně naučit klasifikátor).

Definice 1.11 (Normalizace feature)

Aby nenastával problém např. s odlišnými jednotkami, nebo s různě naškálovanými featury, normalizuje se vydělením referenční hodnotou, respektive různými statistickými metodami, např. standardizací ($\tilde{x}_i = \frac{x_i - \mu}{\sigma}$), nebo 3σ škálováním ($\tilde{x}_i = \frac{x_i - \mu}{3\sigma} + 1$).

Poznámka

Rozhodovací stromy (a náhodné lesy), naivní bayesova metoda atd. nepotřebují normalizaci.

Definice 1.12 (Výběr featur)

Některé featury mohou být totožné, některé zas redundantní.

Takže vybereme nějakou podmnožinu featur a vyzkoušíme. Nebo se naopak podíváme na jednotlivé, ohodnotíme je a vybereme K nejlepších. Další možnost je přidávat je po jedné a testovat je ne samotné, ale s již vybranými. Nebo můžeme začít se všemi a odstraňovat nejhorší (zase oběma způsoby, sekvenčním i jednokrokovým K). Existuje i kombinovaný, který udělá nejdříve jedno a pak druhé. Pak existují i genetické a další algoritmy.

Podle čeho měřit: konzistence (jestli shodné hodnoty jsou ve shodné třídě, viz vzorec v prezentaci), nezávislost na ostatních featurách (opak tzv. korelace) + korelace s třídami, množství informace ($\mathcal{I} = -\log(P(A = a_i))$, $E(\mathcal{I}) = -\sum P(A = a) \cdot \log_2(P(A = a))$), co nám dá, vzdálenost mezi třídami po použití dané featury, ...

Definice 1.13 (Transformace featur)

Unsupervised (minimalizována je ztráta informací): Principal Component Analysis (PCA), Latent Semantic Indexing (LSI), Independent Component Analysis (ICA), ...

Supervised (maximalizuje se vzdálenost mezi třídami): Linear Discriminant Analysis (LDA), Canonical Correlation Analysis (CCA), Partial Least Squares (PLS), ...

Definice 1.14 (PCA)

Také Kaurhunen-Loeve (K-L) method. Hledá v rotacích a deformacích os největší varianci ($b_1^T \Sigma b_1$, kde b_1 je vektor projekce, Σ je matice kovariance). Maximum se najde Lagrangeovy multiplikátory jako místo, kde $b_1^T \Sigma b_1 = \lambda$, kde λ je vlastní číslo Σ , je největší. Pro druhý vektor dostaneme totéž.

$$\begin{aligned}\Sigma &= \frac{1}{N} X X^T (X = \text{rozdíl od průměru}) \\ \Sigma b_j &= \lambda b_j (b_j = \text{vlastní vektory}) \\ X' &= B^T X (B = [b_j])\end{aligned}$$

Definice 1.15 (SVD (singular value decomposition))

Hledáme USV^T tak, že U a V jsou vlastní vektory $A^T A$ a AA^T a V^T je poté diagonální matice vlastních čísel.

$$\begin{aligned}Y &= \frac{1}{\sqrt{N}} X^T \implies Y^T Y = \Sigma \\ Y &= USV^T \\ V &= \text{vlastní čísla matice } Y^T Y = \Sigma\end{aligned}$$

Definice 1.16 (ICA)

Báze nejsou kolmé. Je potřeba aby data byla nezávislá, tedy vycentrujeme odečtením průměru, a vyčistíme tím, že vynásobíme odmocninami vlastních čísel. Počítá se přes entropii (viz prezentace).

Poznámka

Unsupervised transformace featur může vést ke ztrátě oddělení tříd.

Definice 1.17 (LDA)

Snažíme se dostat průměr každé třídy co nejdále od průměru všeho.

Hodně vzorců, viz prezentace. Hledáme tolikarozměrnou projekční nadrovinu (v pre-

zentaci její báze označena w^T), kolik máme tříd - 1.

Poznámka

Lze použít nejdříve LDA, abychom snížili rozměry a následně použít LDA.

Poznámka

Na LDA potřebujeme hodně tréninkových dat. I při hodně dat jsou situace, kdy PCA je lepší (hlavně, když se třídy překrývají).

Definice 1.18 (Rozhodovací hranice (decision boundaries))

Rozděluje prostor na jednotlivé třídy.

Cílem je najít rozhodovací funkci, která bude správně přiřazovat třídy, nebo najít funkci rozumně definující hranice tříd.

Definice 1.19 (Pravděpodobnostní klasifikace)

Mějme M tříd $\{\omega_i\}_{i=1}^M$. Potom pravděpodobnost, že objekt patří do třídy ω_i , za předpokladu, že vektor je \mathbf{x} , je

$$P(\omega_i|\mathbf{x}) = \frac{P(\mathbf{x}|\omega_i) \cdot P(\omega_i)}{\sum_{j=1}^M P(\mathbf{x}|\omega_j) \cdot P(\omega_j)}.$$

Objekt tedy zařadíme do nejpravděpodobnější třídy. Jelikož jmenovatel je všude stejný, tak počítáme

$$\omega_i, \text{ kde } i = \arg \max_j P(\mathbf{x}|\omega_j) \cdot P(\omega_j).$$

Hodí se zvláště, když matice kovariance je diagonální (třídy nejsou kovariantní).

Definice 1.20 (Naivní Bayesův klasifikátor)

Předpokládáme, že každá souřadnice feature vektoru je nezávislá:

$$P(x_1, x_2, \dots, x_D|\omega_i) = P(x_1|\omega_i) \cdot P(x_2|\omega_i) \cdot \dots \cdot P(x_D|\omega_i).$$

Tedy vytvoříme frekvenční tabulku tříd v závislosti na jednotlivých dimenzích a spočítáme pravděpodobnosti. Může se nám ale stát, že při mnoha dimenzích už bude násobení tolika pravděpodobností mimo přesnost, proto se používají logaritmy pravděpodobností.

Navíc pokud nemáme dostatek dat, snadno se nám může (na „nespojité dimenzi“) stát, že pravděpodobnost nějaké třídy bude nula, tedy „přidáme“ 1 ke všem výskytům ve frekvenční tabulce.

Abychom odhadli pravděpodobnost „spojité proměnné“, používáme standardní funkce

hustoty, tedy např. (\bar{x}_{ij} je průměr vzorku, σ_{ij}^2 je variance vzorku):

$$P(x_i = x | \omega_j) = \frac{1}{\sigma_{ij} \sqrt{2\pi}} e^{\frac{-(x - \bar{x}_{ij})^2}{2\sigma_{ij}^2}}.$$

Pokud máme rozložení, kde jsou třeba 2 „hrby“, můžeme rozdělit prvky do „podtříd“ a rovnici výše doplnit o sumu a váhy, které se sečtou na 1 (GMM):

$$P(x_i = x | \omega_j) = \sum_m \frac{w_m}{\sigma_{ijm} \sqrt{2\pi}} e^{\frac{-(x - \bar{x}_{ijm})^2}{2\sigma_{ijm}^2}}.$$

Na spočítání rozdělení do podtříd a vah se používá metoda EM (expectation, Maximization), kde se několikrát zopakuje rozdělení na základě aktuálních vah (v podstatě klasifikaci) a updatování průměru a variance.

2 Hodnocení klasifikátoru

Definice 2.1 (Matice zaměňování - binární klasifikace)

4 hodnoty TP, FN, FP, TN (True / False = klasifikováno správně nebo špatně, Positive / Negative = klasifikováno pozitivně nebo negativně).

$Accuracy = \frac{TP+TN}{P+N}$ má problém, když není dataset rovnoměrný (např. klasifikátor klasifikuje vždy negativně a v setu je jen 1 pozitivní, tak je accuracy 99%). Místo toho lze použít $Errorrate = 1 - Accuracy$ a jiné „rates“, např. $Recall = \frac{TP}{P}$ a $Precision = \frac{TP}{TP+FP}$ (používají se pro sledování thresholdu v grafu křivky Recall – Precision), $truepositivrate = TPR = \frac{TP}{P}$ a $FPR = \frac{FP}{N}$ (dohromady tzv. ROC křivka).

Definice 2.2 (F skóre)

Zkombinovaný recall a precision, má 1 parametr (t je threshold)

$$F_\beta(t) = \frac{(1 + \beta^2)p(t)r(t)}{r(t) + \beta^2 p(t)}.$$

Pro balancovaný model chceme maximalizovat F_1 .

Definice 2.3 (Loss funkce)

Znázorňuje cenu za špatné rozhodnutí.

Zero-one loss je standardní loss funkce (1, když klasifikoval špatně, 0 když dobře).