EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Strukturní klasifikátory a jejich aplikace v počítačovém vidění

Učební texty k semináři

Autoři:

Ing. Vojtěch Franc, Ph.D.

Datum:

17. 12. 2012

Centrum pro rozvoj výzkumu pokročilých řídicích a senzorických technologií
CZ.1.07/2.3.00/09.0031

TENTO STUDIJNÍ MATERIÁL JE SPOLUFINANCOVÁN EVROPSKÝM SOCIÁLNÍM FONDEM A STÁTNÍM ROZPOČTEM ČESKÉ REPUBLIKY

Obsah

Kapitola 1

Úvod

Čím se zabývá obor statistického rozpoznávání. Rozpoznávání se zabývá metodami pro odhadovaní skrytých stavů zkoumaného objektu. Zkoumaný objekt se charakterizuje dvěma stavy: pozorovaným stavem $\boldsymbol{x} \in \mathcal{X}$ a skrytým stavem $\boldsymbol{y} \in \mathcal{Y}$. Symbol $\mathcal{X}$ značí množinu všech možných pozorování a symbol $\mathcal{Y}$ množinu všech možných skrytých stavů. Jednou ze základních úloh rozpoznávání je nalézt rozhodovací funkci $h\colon \mathcal{X} \to \mathcal{Y}$, která na základě pozorování objektu $\boldsymbol{x} \in \mathcal{X}$ odhadne jeho skrytý stav $\boldsymbol{y} \in \mathcal{Y}$. V případě, že je množina skrytých stavů $\mathcal{Y}$ konečná, úloha se označuje jako klasifikační a rozhodovací funkci h se říká klasifikátor.
Ve většině praktických úloh, ve kterých se rozpoznávání používá, nelze vztah mezi pozorovaným $\boldsymbol{x}$ a skrytým stavem $\boldsymbol{y}$ popsat deterministicky. Proto se v praxi častěji používají metody statistického rozpoznávání, které modelují pozorovaný a skrytý stav objektu jako realizaci náhodného procesu s danou hustotou pravděpodobnosti $p(\boldsymbol{x}, \boldsymbol{y})$.

Co to je strukturní rozpoznávání. Metody strukturního rozpoznávání se používají v těch případech, kdy je vhodné modelovat zkoumaný objekt jako množinu jednodušších objektů (částí) a vztahů mezi nimi, kde tyto vztahy tvoří nějakou jasnou strukturu.
V praxi se pozorovaný stav $\boldsymbol{x}$ často skládá z několika měření $\boldsymbol{x} = (x_1, \ldots, x_n)$ provedených na objektu. Stejně tak skrytý stav $\boldsymbol{y}$ je často tvořen sekvencí skrytých stavů $\boldsymbol{y} = (y_1, \ldots, y_m)$. Nechť $\mathcal{I} = \{1, \ldots, n\}$ a $\mathcal{J} = \{1, \ldots, m\}$ jsou množiny indexů jednotlivých pozorování respektive skrytých stavů. Doménou strukturního rozpoznávání jsou ty aplikace, kdy množiny $\mathcal{I}$ a $\mathcal{J}$ mají nějakou jasnou strukturu. Příkladem může být rozpoznávání časových řad, kdy množiny $\mathcal{I}$ a $\mathcal{J}$ tvoří řetězec a nebo rozpoznávání obrazu, kdy jsou množiny $\mathcal{I}$ a $\mathcal{J}$ uspořádány do dvourozměrné mřížky.
Naproti tomu klasické "nestrukturní" metody rozpoznávání typicky neberou při modelování zkoumaného objektu strukturu množin $\mathcal{I}$ a $\mathcal{J}$ v úvahu.

Kde se strukturní rozpoznávání používá. Metody strukturního rozpoznávání našly své uplatnění v mnoha oborech jako je například analýza přirozeného

jazyka, automatický překlad textů, bioinformatika, rozpoznávání řeči, počítačová bezpečnost nebo počítačové vidění.
V kapitole 2 popíšeme některé strukturní klasifikátory používané v rozpoznávání obrazu. Konkrétně odstavec 2.1 popisuje strukturní klasifikátor používaný pro segmentaci obrazu. V odstavci 2.2 ukážeme jak lze současně segmentovat a rozpoznávat text v obraze pomocí jednoho strukturního klasifikátoru. V odstavci 2.3 zmíníme aplikaci strukturních klasifikátorů při detekci významných bodů na lidské tváři.

Co to je učení klasifikátorů. Strukturní klasifikátor $h\colon \mathcal{X} \to \mathcal{Y}$ je často dost složitá funkce, kterou není snadné zkonstruovat. Naproti tomu bývá schůdnější sesbírat množinu trénovacích příkladů $\mathcal{D} = \{(\boldsymbol{x}^1, \boldsymbol{y}^1), \ldots, (\boldsymbol{x}^m, \boldsymbol{y}^m)\} \in (\mathcal{X} \times \mathcal{Y})^m$, která obsahuje příklady dvojic pozorování a jejich odpovídajících skrytých stavů. Na základě trénovacích příkladů $\mathcal{D}$ pak lze ve zvolené množině klasifikátorů $\mathcal{H}$ hledat takový, který v nějakém smyslu nejlépe řeší danou klasifikační úlohu. Procesu výběru klasifikátoru na základě trénovacích příkladů se říká učení klasifikátoru.
V kapitole 3 popíšeme Bayesovskou formulaci klasifikační úlohy, která pokrývá velkou třídu reálných aplikací. Dále v této kapitole nastíníme dva základní přístupy jak tuto Bayesovskou úlohu řešit přibližně pomocí generativních metod učení (odstavec 3.2) a diskriminativních metod učení (odstavec 3.3). V posledních několika letech získaly velkou popularitu především diskriminativní metody učení lineárních strukturních klasifikátorů. Konkrétně metoda s anglickým názvem "Structured Output Support Vector Machines" (dále v textu označovaná zkratkou SO-SVM), kterou popíšeme v odstavci 3.4. Mezi výhody algoritmu SO-SVM patří to, že problém učení převádí na úlohu konvexní optimalizace, kterou lze efektivně řešit.
V poslední kapitole (kapitola 4) popíšeme jednoduchý a velmi modulární algoritmus, který specifické konvexní optimalizační úlohy, jež se v učení strukturních klasifikátorů objevují, efektivně řeší.

Literatura. Zájemcům o statistické a strukturní rozpoznávání lze vřele doporučit české vydání knihy [13], z které autor tohoto textu také čerpal.

Kapitola 2

Příklady strukturních klasifikátorů

2.1 Segmentace obrazu

Cílem segmentace obrazu je přiřadit každý pixel vstupního obrázku do jedné z K tříd. Obrázek 2.1 ukazuje příklady segmentace pro různě definované třídy. K segmentaci obrazu můžeme přistupovat jako k úloze strukturní klasifikace. Typickým příkladem strukturního klasifikátoru, který se při segmentaci obrázků používá, je takzvaný "max-sum" klasifikátor popsaný v následujícím odstavci.

2.1.1 Max-sum klasifikátor

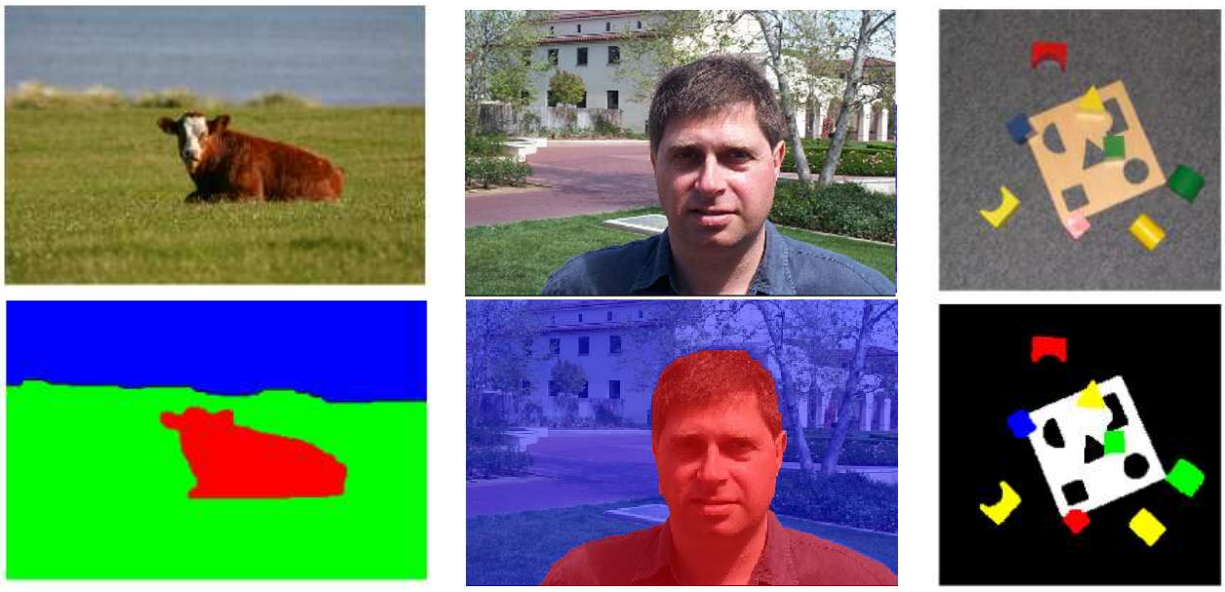
Nechť $(\mathcal{T}, \mathcal{E})$ je neorientovaný graf, kde $\mathcal{T}$ je konečná množina objektů a $\mathcal{E} \subseteq \binom{\mathcal{T}}{2}$ je množina dvojic objektů, které spolu sousedí. Nechť je každý objekt $t \in \mathcal{T}$ charakterizován pozorováním x_t a skrytou proměnnou y_t, které nabývají hodnot z množiny $\mathcal{X}_t$, respektive $\mathcal{Y}_t$. Označme uspořádanou $|\mathcal{T}|$-tici pozorování písmenem $\boldsymbol{x} = (x_t \in \mathcal{X}_t \mid t \in \mathcal{T}) \in \mathcal{X}$ a uspořádanou $|\mathcal{T}|$-tici skrytých proměnných písmenem $\boldsymbol{y} = (y_t \in \mathcal{Y}_t \mid t \in \mathcal{T}) \in \mathcal{Y}$. Každému objektu $t \in \mathcal{T}$ je přiřazena funkce $q_t \colon \mathcal{X}_t \times \mathcal{Y}_t \times \mathbb{R}^n \to \mathbb{R}$. Hodnota funkce $q_t(x_t, y_t; \boldsymbol{\theta})$ určuje kvalitu skrytého stavu y_t pokud je pozorování x_t. Každé dvojici objektů $\{t, t'\} \in \mathcal{E}$ je přiřazena funkce $g_{tt'} \colon \mathcal{Y}_t \times \mathcal{Y}_t \times \mathbb{R}^n \to \mathbb{R}$. Hodnota $g_{tt'}(y_t, y_{t'}; \boldsymbol{\theta})$ určuje kvalitu skrytých stavů y_t a $y_{t'}$. Funkce $q_t(x_t, y_t; \boldsymbol{\theta})$ a $g_t(y_t, y_{t'}; \boldsymbol{\theta})$ jsou parametrizované vektorem $\boldsymbol{\theta} \in \mathbb{R}^n$. Max-sum klasifikátor $h \colon \mathcal{X} \times \mathbb{R}^n \to \mathcal{Y}$ parametrizovaný vektorem $\boldsymbol{\theta}$ vrací pro zadané pozorování $\boldsymbol{x}$ skryté stavy $\boldsymbol{y}$, které mají největší kvalitu:

$$h(\boldsymbol{x}; \boldsymbol{\theta}) = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) \tag{2.1}$$

kde

$$f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) = \sum_{t \in \mathcal{T}} q_t(x_t, y_t; \boldsymbol{\theta}) + \sum_{\{t,t'\} \in \mathcal{E}} g_{tt'}(y_t, y_{t'}; \boldsymbol{\theta}) \,.$$

Hodnota skórovací funkce $f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta})$ určuje shodu mezi pozorováním $\boldsymbol{x}$ a skrytým stavem $\boldsymbol{y}$. Výpočet výstupu max-sum klasifikátoru, tj. ohodnocení pravé strany výrazu (2.1) se označuje jako max-sum problém. Obecný max-sum problém patří do třídy NP-těžkých úloh. Nicméně existují podtřídy



(a) (b) (c)

Obrázek 2.1: Příklady vstupních obrázků (horní řádek) a jejich segmentací (dolní řádek) pro různě definované třídy. V obráku (a) segmentujeme do tříd kráva, tráva a obloha, v obrázku (b) jsou třídy pozadí, objekt a v obrázku (c) se obrázek segmentuje na oblast pozadí a oblasti patřící jednotlivým barevných částem hračky.

max-sum problému, které lze řešit v polynomiálním čase. Například pokud množina $(\mathcal{T}, \mathcal{E})$ je acyklický graf nebo pokud funkce $g_{tt'}$ patří do třídy takzvaných submodulárních funkcí [14].
Max-sum klasifikátor má jasnou statistickou interpretaci. Předpokládejme, že pozorování $\boldsymbol{x}$ a skrytý stav $\boldsymbol{y}$ jsou realizací náhodných veličin $\tilde{\boldsymbol{X}} = (\tilde{X}_t \mid t \in \mathcal{T})$ a $\tilde{\boldsymbol{Y}} = (\tilde{Y}_t \mid t \in \mathcal{T})$. Dále předpokládejme, že pouze proměnné $(\tilde{X}_t, \tilde{Y}_t)$, $t \in \mathcal{T}$ a $(\tilde{Y}_t, \tilde{Y}_{t'})$, $\{t, t'\} \in \mathcal{E}$ jsou na sobě přímo statisticky závislé. Pak je sdružená hustota pravděpodobnosti náhodných veličin $\tilde{\boldsymbol{X}}$ a $\tilde{\boldsymbol{Y}}$ Gibbsovské rozdělení

$$p(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) \,, \tag{2.2}$$

kde $Z(\boldsymbol{\theta})$ je normalizační konstanta. Optimální bayesovský klasifikátor, který minimalizuje pravděpodobnost chybné klasifikace $E_{p(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta})}[\![h(\boldsymbol{x}) \neq \boldsymbol{y}]\!]$, vrací skrytý stav s maximální aposteriorní pravděpodobností, tj.

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} p(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\theta}) \,. \tag{2.3}$$

Odhad skrytého stavu získaný bayesovským klasifikátorem (2.3) je takzvaný MAP odhad (Maximal A Posteriori estimate). Není těžké ukázat, že max-sum klasifikátor (2.1) a optimální bayesovský klasifikátore (2.3) jsou identické,

pokud jsou pozorování $\boldsymbol{x}$ a skryté stavy $\boldsymbol{y}$ generované náhodným procesem s h.p. (2.2).

2.1.2 Použití max-sum klasifikátoru pro segmentaci

V tomto odstavci popíšeme, jak může vypadat konkrétní instance max-sum klasifikátoru vhodná pro segmentaci obrázků. Nechť má vstupní obrázek rozlišení $H \times W$ pixelů. Obrázek je složen z množiny elementárních objektů (pixelů) $\mathcal{T} = \{(i,j) \in \mathcal{N}^2 \mid 1 \leq i \leq W, 1 \leq j \leq H\}$. Každý pixel $t \in \mathcal{T}$ můžeme charakterizovat jeho jasovou hodnotou $x_t \in \mathcal{X}'$, a skrytým stavem $y_t \in \mathcal{Y}' = \{1, \ldots, K\}$, který říká, do které z K tříd pixel patří. Tj. uvažujeme případ, kdy množina pozorování $\mathcal{X}_t = \mathcal{X}'$ a skrytých stavů $\mathcal{Y}_t = \mathcal{Y}'$ pro všechny pixely $t \in \mathcal{T}$ stejná. Množina jasových hodnot může být v případě barevných RGB obrázků například $\mathcal{X}' = \mathbb{R}^3$. To znamená, že pozorování v pixelu $t \in \mathcal{T}$ je vektor $x_t = (r_t, g_t, b_t)$, jehož složky popisují barvu pixelu v RGB prostoru. Jednotlivé třídy v množině $\mathcal{Y}'$ mají typicky nějaký sémantický význam, např. na obrázku 2.1(a) se segmentuje do $K = 3$ tříd: kráva, tráva a obloha. Celý obrázek jasových hodnot je tedy $|\mathcal{T}|$-tice $\boldsymbol{x} = (x_t \in \mathcal{X}' \mid t \in \mathcal{T}) \in \mathcal{X}$ a segmentace je $|\mathcal{T}|$-tice $\boldsymbol{y} = (y_t \in \mathcal{Y}' \mid t \in \mathcal{T}) \in \mathcal{Y}$.
Abychom mohli použít max-sum klasifikátor 2.1 pro odhad segmentace, je třeba určit strukturu sousedství $\mathcal{E}$. Pro jednoduché obrázky obvykle stačí předpokládat, že každý pixel je přímo závislý jen na svých čtyřech nejbližších sousedech (až na okrajové pixely), tj.

$$\mathcal{E} = \{\{(i,j),(i',j')\} \in \mathcal{T}^2 \mid |i-i'| + |j-j'| = 1\} .$$

Dále musíme konkretizovat tvar funkcí $q_t(x_t, y_t; \boldsymbol{\theta})$ a $g_{tt'}(y_t, y_{t'}; \boldsymbol{\theta})$, z kterých je max-sum klasifikátor postaven. Hodnota funkce $q_t(x_t, y_t; \boldsymbol{\theta})$ může být například určena jako vážená lineární kombinace určitých příznaků spočtených na RGB složkách $x_t = (r_t, g_t, b_t)$, tzn. že

$$q_t(x_t, y_t; \boldsymbol{\theta}) = \langle \boldsymbol{\phi}^q(x_t), \boldsymbol{\theta}_{y_t} \rangle , \tag{2.4}$$

kde $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ značí skalární součin vektorů $\boldsymbol{u}$ a $\boldsymbol{v}$, $\boldsymbol{\phi}\colon \mathcal{X}' \to \mathbb{R}^p$, je zobrazení, které vrací p příznaků spočtených z $x_t = (r_t, g_t, b_t)$, a $\boldsymbol{\theta}^q_{y_t} \in \mathbb{R}^p$ jsou váhy lineární kombinace, pokud je skrytý stav y_t. Z definice (2.4) vyplývá, že funkce $q_t(x_t, y_t; \boldsymbol{\theta})$ jsou pro všechy pixely $t \in \mathcal{T}$ definované stejně. Jednoduché příznaky mohou být například

$$\boldsymbol{\phi}(x_t) = \left(r_t, g_t, b_t, r_t^2, g_t^2, b_t^2, 1\right) .$$

Takto definované funkce byly použity například pro získání segmentace v obrázku 2.1 (a) a (c).

Co se týče funkcí $g_{tt'}(y_t, y_{t'}; \boldsymbol{\theta})$, je často rozumné předpokládat, že jejich tvar je pro všechny dvojice $\{t, t'\} \in \mathcal{E}$ stejný, tj. $g_{tt'}(y_t, y_{t'}; \boldsymbol{\theta}) = g(y_t, y_{t'}; \boldsymbol{\theta})$, $\{t, t'\} \in \mathcal{E}$. Pokud nemáme další apriorní informaci, nemusíme tvar funkce $g(y_t, y_{t'}; \boldsymbol{\theta})$ dále omezovat, tj. můžeme ji reprezentovat jako uspořádanou $|\mathcal{Y}'|^2$-tici čísel

$$\boldsymbol{\theta}^g = (g(y, y') \in \mathbb{R} \mid y \in \mathcal{Y}', y' \in \mathcal{Y}') \,.$$

Tímto jsme max-sum klasifikátor specifikovali až na $n = p|\mathcal{Y}'| + |\mathcal{Y}'|^2$ čísel, reprezetovaných vektorem parametrů

$$\boldsymbol{\theta} = \left((\boldsymbol{\theta}^q_y, y \in \mathcal{Y}'), \boldsymbol{\theta}^g\right) \in \mathbb{R}^n \,.$$

Zbývá vyřešit jak zvolit parametry $\boldsymbol{\theta} \in \mathbb{R}^n$ tak, aby max-sum klasifikátor segmentoval obrázky podle našich představ.
V kapitole 3 popíšeme způsob, jak tyto parametry získat učením z trénovací množiny příkladů. Trénovací množinou se zde myslí množina

$$\{(\boldsymbol{x}^1, \boldsymbol{y}^1), \ldots, (\boldsymbol{x}^m, \boldsymbol{y}^m)\} \in (\mathcal{X} \times \mathcal{Y})^m \,,$$

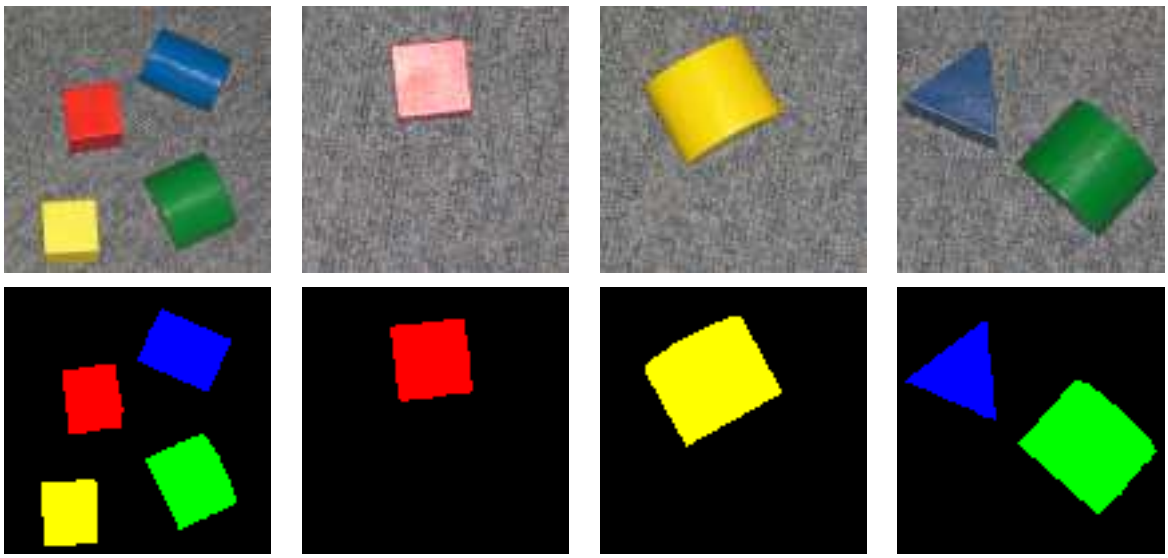
která obsahuje dvojice obrázků, a jejich manuálně získané segmentace. Například pro segmentační problém z Obrázku 2.1(c) může trénovací množina vypadat, jak je ukázáno na Obrázku 2.2.
Metody učení, o kterých budeme mluvit dále, předpokládají, že skórovací funkce strukturního klasifikátoru je lineární v parametrech, které se mají učit, tj.

$$f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) = \langle \boldsymbol{\theta}, \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}) \rangle \,,$$

kde $\boldsymbol{\Psi} \colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^n$ je dané zobrazení z prostoru všech dvojic $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y})$ na prostor parametrů $\mathbb{R}^n$. Není těžké ukázat, že max-sum klasifikátor popsaný výše tento předpoklad splňuje.
Na konec ještě poznamenejme, že popsaná instance max-sum klasifikátoru je dost jednoduchá a myšlená spíše pro ilustraci. Nicméně, uvedený max-sum klasifikátor stačí pro segmentaci jednoduchých obrázků, konkrétně segmentace v Obrázku 2.1 (a) a (c) byly získány přesně tímto modelem.
Další informace o max-sum klasifikátorech a jejich učení se lze dočíst v [5].

2.2 Rozpoznávání a segmentace textu

V tomto odstavci popíšeme strukturní klasifikátor použitelný pro segmentaci a rozpoznávání řádku textu z obraze. Například uvažujme aplikaci, kdy chceme rozpoznávat text na registrační značce automobilu. Obrázek 2.3 ukazuje příklad vstupního obrázku s registrační značkou a požadovaný výstup klasifikátoru.



Obrázek 2.2: Trénovací množina, z které se učí parametry max-sum klasifikátoru pro segmentaci obrazu. Horní řádek jsou vstupní obrázky. Dolní řádek jsou segmentace vytvořené manuálně člověkem.

Obrázek 2.3: Příklad obrázku registrační značky automobilu (horní řádek) a požadovaný výstup strukturního klasifikátoru (dolní řádek), který značku "přečte".

Vstupem klasifikátoru je obrázek $\boldsymbol{x} \in \mathcal{X}$ s rozlišením $H \times W$ pixelů. Obrázek obsahuje řádek textu složený ze známé sady znaků. Klasifikátor vrací segmentaci $\boldsymbol{y} = (s_1, \ldots, s_L) \in \mathcal{Y}$, která rozděluje obrázek $\boldsymbol{x}$ na L segmentů. Každý segment $s = (a, k) \in \mathcal{S}$ popisuje jméno znaku $a \in \mathcal{A}$ a jeho pozici v obrázku $k \in \{1, \ldots, W\}$. Předpokládejme, že každý znak $a \in \mathcal{A}$ má fixní šířku, která je dána funkcí $\omega\colon \mathcal{A} \to \mathcal{N}$. Zaveďme dále funkci $a\colon \mathcal{S} \to \mathcal{A}$, která vrací znak daného segmentu, funkci $k\colon \mathcal{S} \to \mathcal{N}$, která vrací pozici segmentu a funkci $L\colon \mathcal{Y} \to \mathcal{N}$, která vrací počet segmentů. Přípustná segmentace $\boldsymbol{y} \in \mathcal{Y}$ je složena z nepřekrývajících se segmentů, které pokrývají celý vstupní obrázek. To lze formálně zapsat tak, že přípustná segmentace $\boldsymbol{y} \in \mathcal{Y}$ musí splňovat následující podmínky:

$$\begin{aligned} k(s_1) &= 1\,, \\ W &= k(s_L) + \omega(s_L) - 1\,, \\ k(s_i) &= k(s_{i-1}) + \omega(s_{i-1})\,, \qquad \forall i = 1, \ldots, L-1\,. \end{aligned}$$

Každý znak můžeme modelovat etalonem $\boldsymbol{\nu}_a \in \mathbb{R}^{H \times \omega(a)}$, $a \in \mathcal{A}$. Označme písmenem $\boldsymbol{\theta} = (\boldsymbol{\nu}_a, a \in \mathcal{A}) \in \mathbb{R}^n$ vektor, který reprezentuje etalony všech znaků. Pokud etalony znaků známe, pak z nich můžeme vygenerovat syntetický obrázek. Syntetický obrázek je plně popsán segmentací $\boldsymbol{y} \in \mathcal{Y}$ a etalony $\boldsymbol{\theta} \in \mathbb{R}^n$. Podobnost mezi vstupním obrázkem $\boldsymbol{x}$ a syntetickým obrázkem určeným dvojicí $(\boldsymbol{y}, \boldsymbol{\theta})$ můžeme určit např. spočtením jejich korelace

$$f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) = \underbrace{\sum_{i=1}^{L(y)} \sum_{j=1}^{\omega(a(s_i))} \langle \mathrm{col}(\boldsymbol{x}, j + k(s_i) - 1), \mathrm{col}(\boldsymbol{\nu}_{a(s_i)}, j) \rangle}_{\left\langle \text{ULK 68-39}, \text{ULK 68-39} \right\rangle} \quad , \tag{2.5}$$

kde $\mathrm{col}(\boldsymbol{x}, i)$ značí i-tý sloupec obrázku $\boldsymbol{x}$. Vstupní obrázek $\boldsymbol{x}$ můžeme segmentovat tak, že mezi všemi přípustnými syntetickým obrázky vybereme takový, který je mu nejpodobnější a jeho segmentaci prohlásíme za segmentaci vstupního obrázku $\boldsymbol{x}$. Tj. strukturní klasifikátor bude počítat

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta})\,. \tag{2.6}$$

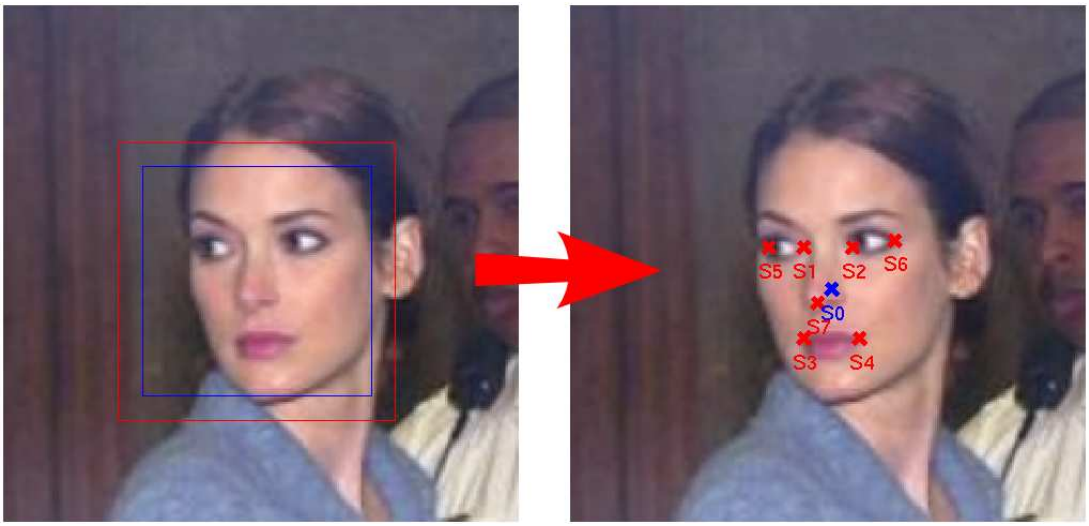
Z výstupu strukturního klasifikátoru (2.6) můžeme vyčíst jak pozice znaků $k(\hat{s}_i)$, tak jejich jména $a(\hat{s}_i)$. Podobně jako v případě strukturního klasifikátoru pro segmentaci obrazu, i skórovací funkce (2.5) je lineární v parametrech $\boldsymbol{\theta}$, tj. můžeme zkonstruovat zobrazení $\boldsymbol{\Psi}\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^n$ takové, že

$$f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) = \langle \boldsymbol{\theta}, \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}) \rangle\,.$$

K tomu aby šlo klasifikátor (2.6) použít v praxi, je třeba určit etalony $\boldsymbol{\theta}$. Toho lze opět dosáhnout pomocí metod učení z příkladů, kdy trénovací množina bude sestávat z příkladů vstupních obrázků obsahující značky a jejich manuálně získané segmentace.
Nakonec opět poznamenejme, že v praxi je třeba uvedený strukturní model dále zesložitit. Například, je téměř vždy nutné modelovat měřítko znaků (zde jsme uvažovali, že šířka znaků je známá). Dále porovnávání obrázků na základě počítání korelací jejich jasových hodnot funguje dobře jen v jednoduchých případech. V praxi je třeba místo jasových hodnot korelovat sofistikovanější příznaky počítané na jasových hodnotách.

2.3 Detekce významných bodů na lidské tváři

Další problém, který se daří řešit pomocí strukturní klasifikace je detekce významných bodů na lidské tváři. Významné body na lidské tváři jsou například koutky očí, koutky úst či špička nosu. Odhad pozice významných bodů je nezbytná součást většiny systémů pro automatické rozpoznávání tváří. Obrázek 2.4 ukazuje příklad vstupu a výstupu takového detektoru.
Detektor významných bodů lze efektivně implementovat strukturním klasifikátorem, který k popisu tváře používá tzv. "Deformable Part Model" (DPM) [4]. DPM modeluje rozpoznávaný objekt jako množinu částí a geometrických omezení mezi vybranými dvojicemi částí. Části mohou být např. lokalní modely očí a geometrické omezení může např. udávat jejich přípustné pozice. Lokální modely částí tváře, stejně jako tvar geometrických omezení, lze učit z příkladů. Konkrétní implementace detektoru významných bodů a algoritmu jeho učení je popsaná v článku [21]. Tento model a algoritmus učení probereme podrobně na semináři.



Obrázek 2.4: Příklad vstupu (levý obrázek) a výstupu (pravý obrázek) strukturního klasifikátoru, který detekuje významné body na lidské tváři.

Kapitola 3

Učení klasifikátorů

V tomto odstavci formulujeme obecnou bayesovskou úlohu rozpoznávání, která pokrývá velkou třídu aplikací. V dalších odstavcích pak popíšeme metody jejího řešení pomocí algoritmů učení. Nakonec kapitoly popíšeme konkrétní instanci učícího algoritmu.

3.1 Formulace úlohy

Zkoumaný objekt se charakterizuje dvěma stavy. Pozorovaným stavem $\boldsymbol{x} \in \mathcal{X}$ a skrytým stavem $\boldsymbol{y} \in \mathcal{Y}$. Symbol $\mathcal{X}$ značí množinu všech možných pozorování a symbol $\mathcal{Y}$ značí konečnou množinu všech možných skrytých stavů. Předpokládejme, že pozorování $\boldsymbol{x}$ a skrytý stav $\boldsymbol{y}$ je realizací nějakého náhodného procesu s h.p. $p(\boldsymbol{x}, \boldsymbol{y})$. Cílem je nalézt klasifikátor $h\colon \mathcal{X} \to \mathcal{Y}$, který na základě pozorování objektu $\boldsymbol{x} \in \mathcal{X}$ odhadne jeho skrytý stav $\boldsymbol{y} \in \mathcal{Y}$. Predikce klasifikátoru se hodnotí pomocí ztrátové funkce $\Delta\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. Číslo $\Delta(\boldsymbol{y}, \boldsymbol{y}')$ je penalta, kterou platíme, pokud je skutečný skrytý stav $\boldsymbol{y}$ a klasifikátor odpoví $\boldsymbol{y}'$. Daný klasifikátor h se charaterizuje matematickým očekáváním hodnoty ztrátové funkce

$$R_{\text{Bayes}}(h) = \sum_{\boldsymbol{x} \in \mathcal{X}} \sum_{\boldsymbol{y} \in \mathcal{Y}} p(\boldsymbol{x}, \boldsymbol{y}) \Delta(\boldsymbol{y}, h(\boldsymbol{x})) \,. \tag{3.1}$$

Číslo $R_{\text{Bayes}}(h)$ se nazývá Bayesovský risk klasifikátoru h. Bayesovská úloha spočívá v nalezení bayesovského klasifikátoru, tj. klasifikátoru, který má minimální bayesovský risk. Z (3.1) je vidět, že bayesovský klasifikátor je dán vztahem

$$h^*(\boldsymbol{x}) = \operatorname*{argmin}_{\boldsymbol{y}' \in \mathcal{Y}} \sum_{\boldsymbol{y} \in \mathcal{Y}} p(\boldsymbol{y} \mid \boldsymbol{x}) \Delta(\boldsymbol{y}, \boldsymbol{y}') \,. \tag{3.2}$$

Řešení bayesovské úlohy vyžaduje znalost statistického modelu $p(\boldsymbol{y} \mid \boldsymbol{x})$, který není téměř nikdy k dispozici. Místo toho bývá schůdné sesbírat množinu příkladů $\mathcal{D} = \{(\boldsymbol{x}^1, \boldsymbol{y}^1), \ldots, (\boldsymbol{x}^m, \boldsymbol{y}^m)\} \in (\mathcal{X} \times \mathcal{Y})^m$, o které předpokládáme, že obsahuje realizace statisticky nezávislých náhodných veličin s rozdělením $p(\boldsymbol{x}, \boldsymbol{y})$. V dalších dvou odstavcích zmíníme dva základní přístupy jak řešit bayesovskou úlohu přibližně s použitím trénovací množiny příkladů $\mathcal{D}$.

Nakonec je na místě zmínit fakt, že ani přesná znalost modelu $p(\boldsymbol{x}, \boldsymbol{y})$ nemusí vždy znamenat jednoduchou úlohu. Problém může být i samotné ohodnocení bayesovského klasifikátoru (3.2), které ve strukturním rozpoznávání často vede na problém neřešitelný v polynomiálním čase a je třeba používat různé aproximativní metody řešení. Např. ohodnocení max-sum klasifikátoru (odstavec 2.1.1) je obecně NP-těžká úloha zatímco klasifikátor pro segmentaci a rozpoznavání textu (odstavec 2.2) lze řešit pomocí dynamického programování.

3.2 Generativní metody

Generativní metody se snaží využít trénovací množinu $\mathcal{D}$ k nalezení statistického modelu $\hat{p}(\boldsymbol{x}, \boldsymbol{y})$, který co nejlépe aproximuje skutečný model $p(\boldsymbol{x}, \boldsymbol{y})$. Tento krok většinou vyžaduje dost podrobnou znalost o tom, jak jsou data generovaná. Typicky se postupuje tak, že se na základě zkoumání problému navrhne třída statistických modelů $\mathcal{P}$, která by měla obsahovat alespoň jeden model blízký tomu skutečnému. Konkrétní aproximace statistického modelu se pak učí (dhaduje) z trénovacích dat $\mathcal{D}$ pomocí metody maximální věrohodnosti, tj. učení vede na problém

$$\hat{p} = \operatorname*{argmax}_{p' \in \mathcal{P}} \prod_{i=1}^{m} p'(\boldsymbol{x}^i, \boldsymbol{y}^i) \,.$$

Naučený statistický model $\hat{p}(\boldsymbol{x}, \boldsymbol{y})$ se následně použije jako náhrada za skutečný model $p(\boldsymbol{x}, \boldsymbol{y})$ v definici bayesovského klasifikátoru (3.2).

3.3 Diskriminativní metody

Diskriminativní metody hledají klasifikátor přímo bez nutnosti odhadovat statistický model $p(\boldsymbol{x}, \boldsymbol{y})$. Místo popisování procesu generujícího data může být jednodušší přímo modelovat bayesovský klasifikátor (3.2). Například můžeme navrhnout třídu klasifikátorů $\mathcal{H}$, která obsahuje alespoň jeden takový, jenž dostatečně dobře aproximuje bayesovský klasifikátor (3.2). Konkrétní klasifikátor vybíráme na základě nějaké rozumné aproximace bayesovského risku $R_{\text{Bayes}}(h)$, kterou lze ohodnotit bez znalosti statistického modelu $p(\boldsymbol{x}, \boldsymbol{y})$. Bayesovský risk $R_{\text{Bayes}}(h)$ lze aproximovat například empirickým riskem

$$R_{\text{emp}}(h) = \frac{1}{m} \sum_{i=1}^{m} \Delta(\boldsymbol{y}^i, h(\boldsymbol{x}^i)) \,,$$

který měří průměrnou hodnotu ztrátové funkce na trénovací množině $\mathcal{D}$. Učení klasifikátoru pak vede na úlohu minimalizace empirického risku

$$\hat{h} = \operatorname*{argmin}_{h \in \mathcal{H}} R_{\text{emp}}(h) \,. \tag{3.3}$$

O konkrétní implementaci toho principu budeme mluvit podrobně v odstavci 3.4, proto uveďme jaké jsou obecně výhody a úskalí tohoto přístupu:

+ Odhadovat klasifikátor přímo bývá často jednodušší úkol, než odhadovat plný statistický model $p(\boldsymbol{x}, \boldsymbol{y})$ jako se to dělá v generativním učení.

+ Ukazuje se, že chyba, které se dospustíme při specifikaci třídy klasifikátorů $\mathcal{H}$, má na výslednou přesnost klasifikátoru menší vliv, než chyba při specifikaci třídy statistických modelů $\mathcal{P}$.

- Složitost úlohy (3.3) závisí jak na volbě (zvolené parametrizaci) třídy klasifikátorů $\mathcal{H}$, z které se vybírá, tak na volbě ztrátové funkce Δ. Bohužel, pro většinu ztrátových funkcí používaných v klasifikaci (a to i těch nejjednodušších) neexistuje efektivní algoritmus, který by úlohu (3.3) řešil přesně a tudíž se používájí různé proximace.

- Nízká hodnota emprického risku $R_{\text{emp}}(h)$ obecně neznamená nízkou hodnotu bayesovského risku $R_{\text{Bayes}}(h)$ bez ohledu na to, kolik máme trénovacích příkladů. Tomuto jevu se říká problém generalizace a je předmětem teorie statistického učení [22]. Praktickým důsledkem je, že kromě minimalizace empirického risku je během učení třeba kontrolovat i složitost třídy klasifikátorů $\mathcal{H}$.

- Narozdíl od generativních metod není obecně jednoduché použít diskriminativní metody pro učení z nekompletních trénovacích dat.

Přes zmíněné problémy existuje velmi efektivní implementace principu diskriminativního učení, která je popsaná v dalším odstavci.

3.4 Structured Output Support Vector Machines

Metoda Structured Output Support Vector Machines (SO-SVM) [20] je diskriminativní metoda pro učení obecných lineárních klasifikátorů. Tj. třída klasifikátorů $\mathcal{H}$, ze které se během učení vybíra, obsahuje klasifikátory $h\colon \mathcal{X} \times \mathbb{R}^n \to \mathcal{Y}$ popsané vztahem

$$h(\boldsymbol{x}; \boldsymbol{\theta}) = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} \langle \boldsymbol{\theta}, \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}) \rangle \,, \tag{3.4}$$

kde $\boldsymbol{\Psi}\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^n$ je dané zobrazení a $\boldsymbol{\theta} \in \mathbb{R}^n$ je vektor parametrů. Vektor parametrů $\boldsymbol{\theta}$ je tedy indexem určujícím konkrétní lineární klasifikátor z množiny $\mathcal{H}$.

Metoda SVM aproximuje empirický risk po částech lineární konvexní funkcí

$$R(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \max_{\boldsymbol{y}\in\mathcal{Y}} \Big[\Delta(\boldsymbol{y}, \boldsymbol{y}^i) + \langle \boldsymbol{\theta}, \boldsymbol{\Psi}(\boldsymbol{x}^i, \boldsymbol{y}) - \boldsymbol{\Psi}(\boldsymbol{x}^i, \boldsymbol{y}^i)\rangle\,,\Big]\,, \tag{3.5}$$

což je rozumná aproximace za předpokladu, že pro ztrátovou funkci platí $\Delta(\boldsymbol{y}, \boldsymbol{y}) = 0$ a $\Delta(\boldsymbol{y}, \boldsymbol{y}') \geq 0$, $\forall \boldsymbol{y}, \boldsymbol{y}' \in \mathcal{Y}$. Pak není těžké ukázat, že platí $R_{\text{emp}}(h(\cdot; \boldsymbol{\theta})) \leq R(\boldsymbol{\theta})$, $\boldsymbol{\theta} \in \mathbb{R}^n$. Učení parametrů lineárního klasifikátoru (3.4) pomocí metody SVM se formuluje jako konvexní minimalizační úloha

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}\in\mathbb{R}^n} F(\boldsymbol{\theta}) \quad \text{kde} \quad F(\boldsymbol{\theta}) = \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2 + R(\boldsymbol{\theta})\,. \tag{3.6}$$

Kriteriální funkce $F(\boldsymbol{\theta})$, která nahrazuje bayesovský risk $R_{\text{bayes}}(h)$, je součtem kvadratického regularizačního členu a risku $R(\boldsymbol{\theta})$. Účelem regularizačního členu je omezit prostor parametrů, ze kterých se vybírá, a tím zamezit možnému přefitování klasifikátoru. Síla regularizace se řídí číslem $\lambda > 0$, jehož optimální hodnota se typicky ladí na validační množině příkladů.

Příklady strukturních klasifikátorů uvedené v kapitole (2) byly instancí lineárního klasifikátoru (3.4). K naučení jejich parametrů pomocí metody SO-SVM musíme vyřešit konkrétní instaci konvexního problému (3.6). Řešení tohoto problému je obecně těžké. Protože kriteriální funkce $F(\boldsymbol{\theta})$ není hladká, nelze použít běžné algoritmy hladké optimalizace. Složitost problému vynikne, pokud ho převedeme na ekvivalentní úlohu kvadratického programování:

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}\in\mathbb{R}^n} \Big[\frac{\lambda}{2}\|\boldsymbol{\theta}\|^2 + \frac{1}{m}\sum_{i=1}^{m}\xi_i\Big] \tag{3.7}$$

za omezení

$$\xi_i \;\geq\; \Delta(\boldsymbol{y}, \boldsymbol{y}^i) + \langle \boldsymbol{\theta}, \boldsymbol{\Psi}(\boldsymbol{x}^i, \boldsymbol{y}) - \boldsymbol{\Psi}(\boldsymbol{x}^i, \boldsymbol{y}^i)\rangle\,, \qquad \boldsymbol{y} \in \mathcal{Y}\,, i = 1, \ldots, m\,.$$

Přestože problém konvexního kvadratického programování je dobře prozkoumán, nelze standardní metody k řešení (3.7) použít. Hlavní překážkou je ohromný počet omezujících podmínek. Konkrétně problém (3.7) má $|\mathcal{Y}|m$ omezujících podmínek, kde $\mathcal{Y}$ značí množinu skrytých parametrů a m je počet trénovacích příkladů. V aplikacích strukturní klasifikace typicky roste velikost množiny $|\mathcal{Y}|$ exponenciálně s počtem elementárních objektů $\mathcal{T}$. Například v problému segmentace obrazu je $|\mathcal{Y}| = K^{H\times W}$ počet všech možných segmentací obrazu velikosti $H \times W$ do K tříd. V další kapitole popíšeme jednoduchý optimalizační algoritmus, který většinou řeší problém (3.6) velmi efektivně.

Kapitola 4

Optimalizační algoritmy v učení

Mnoho populárních metod strojového učení je založeno na principu *minimalizace regularizovaného empirického risku* (regularized risk minimization principle [16]). V těchto metodách je problém učení převeden na úlohu *konvexní optimalizace*

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}\in\mathbb{R}^n} F(\boldsymbol{\theta}) \quad \text{kde} \quad F(\mathbf{w}) = \lambda\Omega(\boldsymbol{\theta}) + R(\boldsymbol{\theta})\,. \tag{4.1}$$

Kriteriální funkce $F\colon \mathbb{R}^n \to \mathbb{R}$, označovaná jako *regularizovaný risk*, je součtem *regularizačního členu* $\Omega\colon \mathbb{R}^n \to \mathbb{R}$ a *empirického risku* (nebo jeho aproximace) $R\colon \mathbb{R}^n \to \mathbb{R}$. Funkce Ω a R jsou obě konvexní. Číslo $\lambda \in \mathbb{R}_+$ je zvolená regularizační konstanta a $\boldsymbol{\theta} \in \mathbb{R}^n$ je vektor parametrů, jenž se má učit z dat. Empirický risk R hodnotí, jak dobře parametr $\boldsymbol{\theta}$ vysvětluje trénovací data. Funkcí regularizační členu Ω je snížit pravděpodobnost, že během učení dojde k přefitování. Zatímco Ω bývá typicky jednoduchá funkce, empirický risk R je často definován složitým výrazem, jehož ohodnocení bývá výpočetně velmi náročné.

Příklad 1 *Metoda SO-SVM posaná v odstavci 3.4 vede na optimalizační problém (3.6), jenž je instancí problému (4.1). V případě SO-SVM je* $\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{\theta}\|^2$ *a risk* $R(\boldsymbol{\theta})$ *je dán vzorcem (3.5).*

Kromě SO-SVM existuje dlouhý seznam dalších metod strojového učení, které jsou ve své podstatě algoritmem řešícím speciální instanci problému (4.1), například SVM regrese [22], učení detektorů neobvyklých událostí (novelty detection) [15], algoritmy učení klasifikátorů minimalizujících složité ztrátové funkce [9], logistická regrese [3] a nebo učení Gaussovských procesů [23].
Pro praktické využití zmíněných metod strojového učení je nezbytné, aby pro danou instanci problému (4.1) existoval efektivní optimalizační algoritmus. Pokud je funkce F diferencovatelná, lze problém (4.1) řešit pomocí standardních metod pro hladkou optimalizaci. Mezi populární algoritmy patří například Newtonova metoda nebo kvazi-newtonovské metody jako LBFGS [12]. V případech, kdy F je nediferencovatelná, bývá problém (4.1) převeden na nějakou známou ekvivalentní úlohu. Například učení SO-SVM klasifikátorů lze

převést na úlohu kvadratického programování (3.7). Bohužel, existující implementace algoritmů pro řešení obecných úloh kvadratického programování nejsou dostatečně efektivní na to, aby byly použitelné pro velké instance optimalizačních úloh, které se běžně vyskytují v praktických aplikacích. Z tohoto důvodu bylo navrženo velké množství specializovaných optimalizačních algoritmů pro minimalizaci konkrétních instancí problému (4.1).
V dalších odstavcích popíšeme obecný algoritmus pro řešení problému (4.1), který byl publikován v práci [18]. Tento algoritmus, nazvaný "Bundle Method for Risk Minimization" (BMRM), je variantou "Bundle methods", které patří mezi klasické metody pro nehladkou optimalizaci [10, 11]. Mezi hlavní přednosti algoritmu BMRM patří jeho obecnost, modularita, snadná paralelizovatelnost a garantovaná konvergence k ε-optimálnímu řešení v $\mathcal{O}(\frac{1}{\varepsilon})$ iteracích. V praktických úlohách bývá rychlost konvergence algoritmu BMRM srovnatelná se specializovanými algoritmy navrženými na míru konkrétním instancím problému (4.1).

4.1 Algoritmus BMRM

Klíčovou myšlenkou algoritmu BMRM je aproximovat *původní problém* (4.1) jednodušším *redukovaným problémem*

$$\boldsymbol{\theta}_t = \operatorname*{argmin}_{\boldsymbol{\theta}\in\mathbb{R}^n} F_t(\boldsymbol{\theta}) \quad \text{kde} \quad F_t(\boldsymbol{\theta}_t) = \lambda\Omega(\boldsymbol{\theta}) + R_t(\boldsymbol{\theta})\,. \tag{4.2}$$

Redukovaný problém (4.2) se z původního problému (4.1) získá tak, že v kriteriální funkci nahradíme empirický risk R jeho po částech lineární aproximací R_t, zatímco regularizační člen Ω zůstane nezměněn. Přibližný model risku R_t je definován vztahem

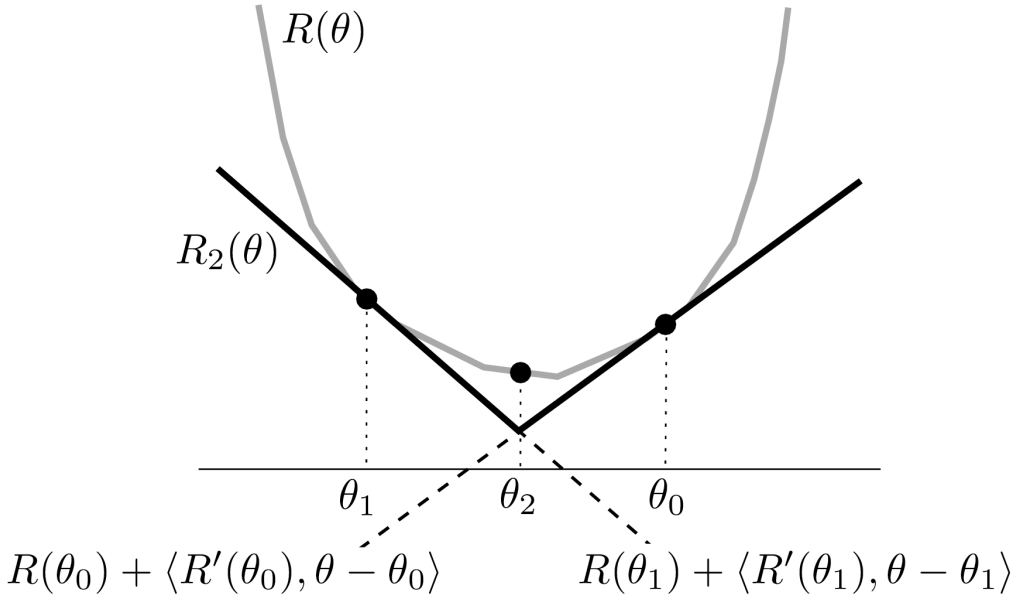
$$R_t(\boldsymbol{\theta}) = \max_{i=0,\ldots,t-1} \left[R(\boldsymbol{\theta}_i) + \langle R'(\boldsymbol{\theta}_i), \boldsymbol{\theta} - \boldsymbol{\theta}_i \rangle \right], \tag{4.3}$$

kde $R'(\boldsymbol{\theta}') \in \mathbb{R}^n$ je libovolný subgradient funkce R v bodě $\boldsymbol{\theta}'$. Připomeňme, že vektor $R'(\boldsymbol{\theta}') \in \mathbb{R}^n$ je subgradientem konvexní funkce R v bodě $\boldsymbol{\theta}'$, pokud platí

$$R(\boldsymbol{\theta}) \geq R(\boldsymbol{\theta}') + \langle R'(\boldsymbol{\theta}'), \boldsymbol{\theta} - \boldsymbol{\theta}' \rangle\,, \qquad \forall \boldsymbol{\theta} \in \mathbb{R}^n\,. \tag{4.4}$$

Subgradient existuje v libovolném bodě $\boldsymbol{\theta}' \in \mathbb{R}^n$, v němž je hodnota konvexní funkce $R(\boldsymbol{\theta}')$ konečná.

Příklad 2 *V případě metody SO-SVM, která má risk daný vztahem (3.5), lze subgradient v bodě* $\boldsymbol{\theta}$ *spočítat z Danskinovy věty (např., Proposition B.25*



Obrázek 4.1: Na obrázku je příklad konvexní funkce $R(\boldsymbol{\theta})$ a jejího přibližného, po částech lineárního modelu $R_2(\boldsymbol{\theta}) = \max\{R(\boldsymbol{\theta}_0)+\langle R'(\boldsymbol{\theta}_0), \boldsymbol{\theta}-\boldsymbol{\theta}_0\rangle, R(\boldsymbol{\theta}_1)+\langle R'(\boldsymbol{\theta}_1), \boldsymbol{\theta}-\boldsymbol{\theta}_1\rangle\}$, který je tvořen dvěma lineárními dolními mezemi, jež aproximují funkci $R(\boldsymbol{\theta})$ přesně v bodech $\boldsymbol{\theta}_0$ a $\boldsymbol{\theta}_1$. Bod $\boldsymbol{\theta}_2$ je minimalizátorem $R_2(\boldsymbol{\theta})$.

v [1]) jako

$$R'(\boldsymbol{\theta}) = \sum_{i=1}^{m} \boldsymbol{\Psi}(\boldsymbol{x}^i, \hat{\boldsymbol{y}}^i) - \sum_{i=1}^{m} \boldsymbol{\Psi}(\boldsymbol{x}^i, \boldsymbol{y}^i)\,, \tag{4.5}$$

kde

$$\hat{\boldsymbol{y}}^i = \operatorname*{argmax}_{\boldsymbol{y}\in\boldsymbol{\mathcal{Y}}} \left[\Delta(\boldsymbol{y}, \boldsymbol{y}^i) + \langle \boldsymbol{\theta}, \boldsymbol{\Psi}(\boldsymbol{x}^i, \boldsymbol{y})\rangle\right]. \tag{4.6}$$

Nerovnice (4.4) definuje lineární dolní mez funkce R a tato mez je v bodě $\boldsymbol{\theta}'$ těsná. Z toho také vyplývá, že i bodové maximum (4.3) definované přes t lineárních dolních mezí je samo po částech lineární dolní mezí funkce R. To znamená, že platí

$$R(\boldsymbol{\theta}) \geq R_t(\boldsymbol{\theta})\,, \forall \boldsymbol{\theta} \in \mathbb{R}^n \qquad \text{a navíc, že} \qquad R(\boldsymbol{\theta}_i) = R_t(\boldsymbol{\theta}_i)\,, i = 0, \ldots, t-1.$$

Z toho je okamžitě vidět, že kriteriální funkce F_t redukovaného problému (4.2) je dolní mezí kriteriální funkce F původního problému (4.1), tj. platí $F(\boldsymbol{\theta}) \geq F_t(\boldsymbol{\theta})\,, \forall \boldsymbol{\theta} \in \mathbb{R}^n$ a $F(\boldsymbol{\theta}_i) = F_i(\boldsymbol{\theta}_i)$, $i = 0, \ldots, t-1$. Řešením redukovaného problému (4.2) získáme kromě přibližného řešení $\boldsymbol{\theta}_t$ i dolní odhad $F_t(\boldsymbol{\theta}_t)$ optimání hodnoty $F(\boldsymbol{\theta}^*)$. Obrázek 4.1 ilustruje lineární dolní mez (4.4) a přibližný model (4.3) na příkladě jednorozměrné konvexní funkce R.

Algorithm 1 Bundle Method for Regularized Risk Minimization (BMRM)

vstup & inicializace: $\varepsilon > 0$, $\boldsymbol{\theta}_0 \in \mathbb{R}^n$, $t \leftarrow 0$
repeat
$t \leftarrow t + 1$
Spočti $R(\boldsymbol{\theta}_{t-1})$ a $R'(\boldsymbol{\theta}_{t-1})$
Aktualizuj model $R_t(\boldsymbol{\theta}) \leftarrow \max_{i=0,\ldots,t-1} R(\boldsymbol{\theta}_i) + \langle R'(\boldsymbol{\theta}_i), \boldsymbol{\theta} - \boldsymbol{\theta}_i \rangle$
Vyřeš redukovaný problém $\boldsymbol{\theta}_t \leftarrow \operatorname{argmin}_{\boldsymbol{\theta}} F_t(\boldsymbol{\theta})$ kde $F_t(\boldsymbol{\theta}) = \lambda\Omega(\boldsymbol{\theta}) + R_t(\boldsymbol{\theta})$
until $F(\boldsymbol{\theta}_t) - F_t(\boldsymbol{\theta}_t) \leq \varepsilon$

BMRM algoritmus 1 začíná z libovolného bodu $\boldsymbol{\theta}_0 \in \mathbb{R}^n$. V dalších iteracích algoritmu se počítá nový bod $\boldsymbol{\theta}_t$ řešením redukovaného problému (4.2). V každé iteraci se přibližný model (4.3) zpřesní přidáním jedné nové lineární dolní meze (4.4), která aproximuje risk R v bodě aktuálního řešení $\boldsymbol{\theta}_t$. Algoritmus končí, pokud je rozdíl mezi hodnotou $F(\boldsymbol{\theta}_t)$ a $F_t(\boldsymbol{\theta}_t)$ menší než zadané $\varepsilon > 0$. Jelikož $F(\boldsymbol{\theta}_t)$ a $F_t(\boldsymbol{\theta}_t)$ jsou horním a dolním odhadem optimání hodnoty $F(\boldsymbol{\theta}^*)$, splnění ukončující podmínky zaručuje, že $F(\boldsymbol{\theta}_t) \leq F(\boldsymbol{\theta}^*) + \varepsilon$.

4.2 Konvergence algoritmu BMRM

Zastavení algoritmu 1 v konečném čase je zaručeno následující větou:

Věta 1 *[19, Theorem 5] Předpokládejme, že platí (i)* $F(\boldsymbol{\theta}) \geq 0$, $\forall \boldsymbol{\theta} \in \mathbb{R}^n$, *(ii)* $\max_{\boldsymbol{g} \in \partial R(\boldsymbol{\theta})} \|\boldsymbol{g}\|_2 \leq G$ *pro všechny* $\boldsymbol{\theta} \in \{\boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_{t-1}\}$, *kde* $\partial R(\boldsymbol{\theta})$ *značí subdiferenciál funkce* R *v bodě* $\boldsymbol{\theta}$ *a (iii)* Ω^* *je dvakrát diferencovatelná funkce s konečnou křivostí, tj.,* $\|\partial^2\Omega^*(\boldsymbol{\mu})\| \leq H^*$ *pro všechny* $\boldsymbol{\mu} \in \{\, \boldsymbol{\mu}' \in \mathbb{R}^t \mid \boldsymbol{\mu}' = \lambda^{-1} A\boldsymbol{\alpha}\ , \|\boldsymbol{\alpha}\|_1 = 1\ , \boldsymbol{\alpha} \geq 0\,\}$, *kde* $\partial^2\Omega^*(\boldsymbol{\mu})$ *je Hessian funkce* Ω^* *v bodě* $\boldsymbol{\mu}$. *Potom se algoritmus 1 zastaví maximálně po*

$$T \leq \log_2 \frac{\lambda F(0)}{G^2 H^*} + \frac{8G^2 H^*}{\lambda\varepsilon} - 1$$

iteracích pro libovolné $\varepsilon < 4G^2H^*\lambda^{-1}$.

Navíc lze ukázat, že v případě, kdy F je dvakrát diferencovatelná a má konečnou křivost, algoritmus 1 vyžaduje maximálně $\mathcal{O}(\log \frac{1}{\varepsilon})$ iterací namísto $\mathcal{O}(\frac{1}{\varepsilon})$ [19, Theorem 5]. Z praktického hlediska je nejvíce omezujícím předpokladem věty (1) požadavek na existenci druhých derivací regularizátoru Ω^*. Tento předpoklad je splněn pro kvadratický regularizátor $\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{\theta}\|_2^2$ používaný v SVM metodách nebo pro regularizátor ve formě negativní entropie $\Omega(\boldsymbol{\theta}) = \sum_{i=1}^n \theta_i \log \theta_i$. Bohužel, uvedený předpoklad neplatí v případě ℓ_1-

normy $\Omega(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_1$, která se často používá k získání řídkého řešení (sparse solution) a výběr příznaků.

4.3 Řešení redukovaného problému

Algoritmus 1 převádí původní problém (4.1) na sérii redukovaných problémů (4.2). Složitost redukovaného problému závisí na počtu proměnných, který je roven dimenzi n vektoru parametrů $\boldsymbol{\theta} \in \mathbb{R}^n$, a na počtu lineárních dolních mezí, který se rovná počtu provedených iterací t. V praxi je t typicky mnohem menší než n. Z tohoto důvodu je výhodné řešit redukovaný problém v jeho duální formě.

Nechť $A = [\boldsymbol{a}_0, \ldots, \boldsymbol{a}_{t-1}] \in \mathbb{R}^{n \times t}$ je matice, jejíž sloupce jsou subgradienty $\boldsymbol{a}_i = R'(\boldsymbol{\theta}_i)$ a nechť $\boldsymbol{b} = [b_0, \ldots, b_{t-1}] \in \mathbb{R}^t$ je sloupcový vektor, jehož komponenty se rovnají $b_i = R(\boldsymbol{\theta}_i) - \langle R'(\boldsymbol{\theta}_i), \boldsymbol{\theta}_i \rangle$. Potom můžeme redukovaný problém (4.2) vyjádřit v jeho ekvivalentní formě

$$\boldsymbol{\theta}_t = \underset{\boldsymbol{\theta} \in \mathbb{R}^n, \xi \in \mathbb{R}}{\operatorname{argmin}} \left[\lambda \Omega(\boldsymbol{\theta}) + \xi \right] \quad \text{za podmínky} \quad \xi \geq \langle \boldsymbol{\theta}, \boldsymbol{a}_i \rangle + b_i \,, \; i = 0, \ldots, t-1 \,. \tag{4.7}$$

Largrangeovský duální problém (viz. např. [2]) k problému (4.7) má tvar [19, Theorem 2]

$$\boldsymbol{\alpha}_t = \underset{\boldsymbol{\alpha} \in \mathbb{R}^t}{\operatorname{argmin}} \left[-\lambda \Omega^*(-\lambda^{-1} A \boldsymbol{\alpha}) + \langle \boldsymbol{\alpha}, \boldsymbol{b} \rangle \right] \quad \text{z.p.} \quad \|\boldsymbol{\alpha}\|_1 = 1 \,, \boldsymbol{\alpha} \geq 0 \,, \tag{4.8}$$

kde $\Omega^* \colon \mathbb{R}^n \to \mathbb{R}^t$ značí konjugovanou funkci k Ω, která je definovaná vztahem

$$\Omega^*(\boldsymbol{\mu}) = \sup \left\{ \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle - \Omega(\boldsymbol{\theta}) \;\middle|\; \boldsymbol{\theta} \in \mathbb{R}^n \right\} .$$

Z optimální hodnoty duálních proměnných $\boldsymbol{\alpha}_t$ lze určit optimální hodnotu primárních proměnných řešením problému $\boldsymbol{\theta}_t = \operatorname{argmax}_{\boldsymbol{\theta} \in \mathbb{R}^n} \left[\langle \boldsymbol{\theta}, -\lambda^{-1} A \boldsymbol{\alpha}_t - \Omega(\boldsymbol{\theta}) \right]$, který se pro diferencovatelné Ω zjednoduší na $\boldsymbol{\theta}_t = \nabla_{\boldsymbol{\mu}} \Omega^*(-\lambda^{-1} A \boldsymbol{\alpha}_t)$.

Příklad 3 (Redukovaný problém pro případ kvadratické regularizace) *V případě kvadratického regularizátoru $\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{\theta}\|_2^2$ používaného v SVM klasifikaci, je konjugovaná funkce rovna $\Omega^*(\boldsymbol{\mu}) = \frac{1}{2}\|\boldsymbol{\mu}\|_2^2$ a duální problém (4.8) má tvar*

$$\boldsymbol{\alpha}_t \in \underset{\boldsymbol{\alpha} \in \mathbb{R}^t}{\operatorname{argmin}} \left[-\frac{1}{2\lambda} \boldsymbol{\alpha}^T A^T A \boldsymbol{\alpha} + \boldsymbol{\alpha}^T \boldsymbol{b} \right] \quad \textit{za podmínky} \quad \|\boldsymbol{\alpha}\|_1 = 1 \,, \boldsymbol{\alpha} \geq 0 \,. \tag{4.9}$$

Primární řešení lze spočítat analyticky podle vzorce $\boldsymbol{\theta}_t = -\lambda^{-1} A \boldsymbol{\alpha}_t$.

4.4 Použití BMRM pro učení strukturních klasifikátorů

Velkou předností algoritmu BMRM je jeho jednoduchost a modularita. K tomu, abychom mohli BMRM použít pro řešení konkrétní instance problému (4.1), stačí implementovat proceduru pro výpočet hodnoty empirického risku $R(\boldsymbol{\theta})$ a jeho subgradientu $R'(\boldsymbol{\theta})$. Samotné jádro algoritmu BMRM, tzn. řešení redukovaného problému, zůstává pro daný regularizátor Ω vždy stejné. Jak bude vypadat použití BMRM pro učení strukturních klasifikátorů pomocí metody SO-SVM ? V tomto případě je regularizační člen kvadratický a redukovaný problém je jednoduchá úloha kvadratrického programování (4.9). Dále jen potřebujeme implementovat dvě procedury. Zaprvé výpočet risku (3.5), a zadruhé výpočet subgradientu (4.5). Při bližším zkoumání zjistíme, že jádrem obou výpočtů je algoritmus, který spočte výstup strukturního klasifikátoru (3.4), respective strukturního klasifikátoru, jehož skórovací funkce je modifikovaná zvolenou ztrátovou funkcí, tj. výpočet (4.6). Tzn. pokud umíme spočítat výstup daného strukturního klasifikátoru, a to bychom měli pokud ho chceme používat, můžeme jeho parametry algoritem BMRM učit bez dalších modifikací.

4.5 Implementace a další modifikace

Implementace algoritmu BMRM je například součástí Shogun toolboxu [17], který sám o sobě obsahuje velkou kolekci dalších algoritmů strojového učení. Urychlenou verzi algoritmu BMRM a její implementaci popisuje např. [6, 7]. Další informace o "cutting plane" metodách a jejich aplikacích ve strojovém učení lze nalézt v [8].

Literatura

[1] Bertsekas, D. P. *Nonlinear Programming.* Athena Scientific, Belmont, MA, 1999.

[2] Boyd, S., Vandenberghe, L. *Convex Optimization.* Cambridge University Press, Cambridge, UK, 2004.

[3] Collins, M., Schapire, R., Singer, Y. Logistic regression AdaBoost and Bregman distance. In *Proceedings of Annual Conference on Computational Learning Theory (COLT)*, s. 158–169. Morgan Kaufman, San Francisco, 2000.

[4] Crandall, D., Felzenszwalb, P., Huttenlocher, D. Spatial priors for part-based recognition using statistical models. In *In CVPR*, s. 10–17, 2005.

[5] Franc, V., Savchynskyy, B. Discriminative learning of max-sum classifiers. *Journal of Machine Learning Research*, sv. 9, č. 1, s. 67–104, January 2008.

[6] Franc, V., Sonnenburg, S. Optimized cutting plane algorithm for large-scale risk minimization. *Journal of Machine Learning Research*, sv. 10, s. 2157–2192, 2010.

[7] Franc, V., Sonnenburg, S. *LIBOCAS.*, 2008 Software available at `http://mloss.org/software/view/85/`.

[8] Franc, V., Sonnenburg, S., Werner, T. *Cutting-Plane Methods in Machine Learning.*, chapter 7, s. 185–218 The MIT Press, Cambridge,USA, 2012.

[9] Joachims, T. A support vector method for multivariate performance measures. In *International Conference on Machine Learning (ICML)*, s. 377–384, 2005.

[10] Kiwiel, K. C. An aggregate subgradient method for nonsmooth convex minimization. *Mathematical Programming*, sv. 27, s. 320–341, 1983.

[11] Lemaréchal, C., Nemirovskii, A., Nesterov, Y. New variants of bundle methods. *Mathematical Programming*, sv. 69, č. 1–3, s. 111–147, 1995.

[12] Nocedal, J., Wright, S. *Numerical Optimization.* Springer Series in Operations Research. Springer, 1999.

[13] Schlesinger, M. I., Hlaváč, V. *Deset přednášek z teorie statistického a strukturního rozpoznávání.* ČVUT, Prague, Czech Republic, 1999.

[14] Schlesinger, M., Flach, B. Some solvable sublcasses of structural recognition problems. In *Czech Pattern Recognition Workshop*. Czech Pattern Recognition Society, 2000.

[15] Schölkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., Platt, J. Support vector method for novelty detection. In Jordan, M. I., Kearns, M. J., Solla, S. A., editors, *Advances in Neural Information Processing Systems 10*, 2000.

[16] Schölkopf, B., Smola, A. *Learning with Kernels.* The MIT Press, MA, 2002.

[17] Sonnenburg, S., Rätsch, G., Henschel, S., Widmer, C., Behr, J., Zien, A., de Bona, F., Binder, A., Gehl, C., Franc, V. The shogun machine learning toolbox. *Journal of Machine Learning Research*, sv. 11, č. 6, s. 1799–1802, June 2010.

[18] Teo, C., Le, Q., Smola, A., Vishwanathan, S. A scalable modular convex solver for regularized risk minimization. In *Proc. Intl. Conf. Knowledge Discovery and Data Mining*, 2007.

[19] Teo, C., Vishwanathan, S., Smola, A., Quoc, V. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, sv. 11, s. 311–365, 2010.

[20] Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, sv. 6, s. 1453–1484, Sep. 2005.

[21] Uřičář, M., Franc, V., Hlaváč, V. Detector of facial landmarks learned by the structured output SVM. In Csurka, G., Braz, J., editors, *VISAPP '12: Proceedings of the 7th International Conference on Computer Vision Theory and Applications*, vol. 1, s. 547–556, Portugal, February 2012. SciTePress — Science and Technology Publications.

[22] Vapnik, V. N. *Statistical learning theory.* Adaptive and Learning Systems. Wiley, New York, New York, USA, 1998.

[23] Williams, C. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In *Learning and Inference in Graphical Models*, s. 599–621. Kluwer Academic, 1998.

Centrum pro rozvoj výzkumu pokročilých řídicích a senzorických technologií
CZ.1.07/2.3.00/09.0031

Ústav automatizace a měřicí techniky
VUT v Brně
Kolejní 2906/4
612 00 Brno
Česká Republika

http://www.crr.vutbr.cz
info@crr.vutbr.cz