Západočeská univerzita v Plzni
Fakulta aplikovaných věd

Aplikované vědy a informatika
Kybernetika a řídící technika

Vypracované otázky ke státní závěrečné zkoušce

(Ing.)

1. června 2017 **Martin Bulín, MSc.**

1 Umělá inteligence [UISZ] 1

1.1 Učící se systémy a klasifikátory [USK] . . . 1
1.1.1 Kritérium minimální chyby. . . . 1
1.1.2 Pravděpodobnostní diskriminační funkce. Souvislost s klasifikátory podle lineární diskriminační funkce, podle nejmenší vzdálenosti, podle nejbližšího souseda a podle k-nejbližšího souseda. . . . 3
1.1.3 Klasifikátor s lineární diskriminační funkcí. Klasifikace do dvou a do více tříd. . . . 6
1.1.4 Metody nastavování klasifikátorů (trénování klasifikátorů). . . . 8
1.1.5 Metody shlukové analýzy (učení bez učitele). . . . 12
1.1.6 Výběr informativních příznaků. . . . 17

1.2 Neuronové sítě [NEU] . . . 18
1.2.1 Základní umělé modely neuronu, vlastnosti, souvislost s biologickým neuronem. . . . 18
1.2.2 Základní typy neuronových sítí. Způsoby činnosti a učení neuronových sítí. 19
1.2.3 Algoritmus backpropagation. . . . 20
1.2.4 Sítě se zpětnou vazbou. Hopfieldova neuronová síť. . . . 22
1.2.5 Samoorganizující se sítě. . . . 24
1.2.6 Oblasti použití neuronových sítí. . . . 26

1.3 Zpracování digitalizovaného obrazu [ZDO] . . . 27
1.3.1 Bodové jasové transformace. . . . 27
1.3.2 Geometrické transformace. . . . 28
1.3.3 Filtrace šumu. . . . 29
1.3.4 Gradientní operátory. . . . 30
1.3.5 Metody segmentace. . . . 31

1.3.6 Matematická morfologie. 33

2 Teorie řízení [TŘSZ] 35

2.1 Lineární systémy 1-2 [LS1], [LS2] 35
2.1.1 Matematické modely spojitých a diskrétních lineárních dynamických systémů. 36
2.1.2 Linearizace nelineárních dynamických systémů, rovnovážné stavy. Harmonická linearizace. 36
2.1.3 Vlastnosti lineárních dynamických systémů. Řiditelnost, pozorovatelnost, kriteria. Vnitřní a vnější stabilita, kriteria. 36
2.1.4 Časové a frekvenční odezvy elementárních členů regulačních obvodů. . . . 36
2.1.5 Základní typy spojitých a diskrétních regulátorů (P,PI,PID, stavové regulátory a stavové regulátory s integračním charakterem), popis, vlastnosti. 36
2.1.6 Struktura regulačních obvodů s jedním a dvěma stupni volnosti, přenosy v regulačním obvodu, princip vnitřního modelu. 36
2.1.7 Problém umístitelnosti pólů a nul nedynamickými a dynamickými regulátory. Požadavky na umístění pólů, konečný počet kroků regulace. . . 36
2.1.8 Požadavky na funkci a kvalitu regulace (přesnost regulace, dynamický činitel regulace, kmitavost, robustnost ve stabilitě a j.), omezení na dosažitelnou kvalitu regulace. 36
2.1.9 Metoda geometrického místa kořenů, pravidla pro konstrukci a využití při syntéze regulátorů, příklady. 36
2.1.10 Přístup k syntéze regulátorů v klasické teorii regulace, klasické metody, heuristické metody. 36
2.1.11 Deterministická rekonstrukce stavu, stavový regulátor s rekonstruktorem stavu. 36
2.1.12 Ljapunovova teorie stability. Ljapunovova rovnice. 36

2.2 Teorie odhadu [TOD] 36
2.2.1 Problémy odhadu, základní etapy vývoje teorie odhadu, náhodné veličiny, náhodné procesy a jejich popis, stochastický systém. 37
2.2.2 Optimální odhad ve smyslu střední kvadratické chyby. Odhad ve smyslu maximální věrohodnosti. 37
2.2.3 Jednorázové a rekurzivní odhady. 37
2.2.4 Odhad stavu lineárního diskrétního systému – filtrace (Kalmanův filtr). . 37
2.2.5 Úlohy odhadu stavu lineárního diskrétního stochastického systému – predikce a vyhlazování. 37
2.2.6 Odhad stavu lineárního systému se spojitým či diskrétním měřením (Kalman-Bucyho filtr). 37

2.3 Optimální systémy [OPS] 37
2.3.1 Optimální programové řízení diskrétních dynamických systémů. Formulace úlohy. Hamiltonova funkce. Nutné podmínky pro optimální řízení. . . 38
2.3.2 Optimální programové řízení spojitých dynamických systémů. Formulace úlohy. Hamiltonova funkce. Nutné podmínky pro optimální řízení. Podmínky transverzality. Pontrjaginův princip minima. 38
2.3.3 Deterministický diskrétní systém automatického řízení. Princip optimality. Bellmanova funkce. Bellmanova optimalizační rekurze. 38

2.3.4 Syntéza optimálního deterministického systému automatického řízení pro diskrétní lineární řízený systém a kvadratické kritérium. Formulace a řešení. Asymptotické řešení a jeho stabilita. 38
2.3.5 Deterministický spojitý systém automatického řízení. Kontinualizace Bellmanovy optimalizační rekurze. 38
2.3.6 Optimální stochastický systém automatického řízení. Strategie řízení. Bellmanova funkce a Bellmanova optimalizační rekurze. 38
2.3.7 Syntéza optimálního systému automatického řízení pro lineární gaussovský řízený systém a kvadratické kritérium. Formulace a řešení. Separační teorém. 38

2.4 Adaptivní systémy [AS] **38**
2.4.1 Základní přístupy k syntéze adaptivních řídicích systémů, schematické vyjádření, srovnání s předpoklady a návrhem standardních regulátorů. . . 39
2.4.2 Adaptivní řízení s referenčním modelem, MIT pravidlo, využití Ljapunovovy teorie stability. 39
2.4.3 Samonastavující se regulátory, charakteristika a základní přístupy k návrhu bloku řízení, přiřazení pólů, diofantické rovnice, minimální variance. . . . 39
2.4.4 Samonastavující se regulátory, charakteristika a základní přístupy k návrhu bloku poznávání, parametrické metody odhadu. 39
2.4.5 Adaptivní systémy na zpracování signálu, adaptivní prediktor, adaptivní filtr, analogie se samonastavujícími se regulátory. 39
2.4.6 Adaptivní řízení a strukturální vlastnost stochastického optimálního řízení, duální řízení, neutralita, separabilita, ekvivalence určitosti. 39

3 Aplikovaná kybernetika [AKSZ] 40

3.1 Umělá inteligence [UI] **40**
3.1.1 Metody řešení úloh v UI 40
3.1.2 Logické formalizmy pro reprezentaci znalostí. Predikátový počet 1. řádu. Rezoluční metoda. 40
3.1.3 Produkční systém. Báze znalostí a báze dat. Dopředné a zpětné šíření. . . 40
3.1.4 Síťové formalizmy pro reprezentaci znalostí. Sémantické sítě. Rámce. Scénáře. 40
3.1.5 Metody hraní her v UI. Procedura minimax, alfa-beta prořezávání. 40

3.2 Modelování a simulace 1 [MS1] **40**
3.2.1 Systém, model, modelování, simulace, systémová analýza. 41
3.2.2 Modelování systému diskrétních událostí, diskrétní simulace. 41
3.2.3 Simulační experiment, studie, analýza rizika, náhoda v simulačních úlohách. 41
3.2.4 Modelování v netechnických oborech (kompartmenty, buněčné automaty, ...). 41
3.2.5 Konstrukce modelů na základě měření, zpracování signálu v časové, frekvenční a časo-frekvenční oblasti, modely periodických procesů. 41
3.2.6 Modely vibrací a kmitání, experimentální modální analýza. 41
3.2.7 Generování náhodných čísel, metoda Monte Carlo a odhad přesnosti simulačních výsledků. 41

3.3 Programové prostředky řízení [PP] **41**
3.3.1 Architektura podnikových řídicích systémů; používané programovací jazyky. 42

3.3.2 Architektura .NET Frameworku; řízený modul, metadata, běh řízeného kódu. 42
3.3.3 Jazyk C Sharp: hodnotové a referenční typy; jednoduché typy, implicitní konverze; výrazy a operátory; příkazy; výjimky. 42
3.3.4 Jazyk C Sharp: Členy a přístup k nim; jmenné prostory; třídy, metody, vlastnosti, konstruktory, destruktory; struktury; pole; delegáty; atributy. . 42
3.3.5 Softwarové komponenty: DLL, RPC, COM; interface; OPC. 42
3.3.6 Operační systémy: procesy a thready, synchronizace, deadlock, inverze priorit; správa paměti; vstupně-výstupní systém, programované vstupy/výstupy, přerušení, DMA, ovladače zařízení; souborové systémy. 42
3.3.7 Operační systémy reálného času: statické a dynamické plánovací algoritmy. 42
3.3.8 Struktury vzdálených a virtuálních laboratoří. 42

3.4 Převodníky fyzikálních veličin [PFV] 42
3.4.1 Struktura a parametry senzorů pro automatizaci, statické a dynamické modely a chyby, metody snižování chyb senzorů. 43
3.4.2 A/D a D/A převodníky, obvody pro úpravu signálů, frekvenční filtry. . . 43
3.4.3 Senzory teploty a tepla, obvody pro měření odporu, kapacity, indukčnosti a frekvence. 43
3.4.4 Senzory polohy a vzdálenosti (odporové, indukční, kapacitní, ultrazvukové, optické). 43
3.4.5 Senzory síly, hmotnosti, deformace, tlaku, rychlosti, zrychlení a vibrací (tenzometrické, piezoelektrické, kapacitní a elektrodynamické). 43
3.4.6 Senzory průtoku, množství, hustoty, viskozity, koncentrace a chemického složení. 43
3.4.7 Elektrické akční členy a jejich budiče (stejnosměrné, střídavé, krokové motory, PWM zesilovače, frekvenční měniče). 43
3.4.8 Hydraulické a pneumatické akční členy (pracovní a řídicí mechanizmy a zdroje tlakového média). 43

Kapitola 1

Umělá inteligence [UISZ]

1.1 Učící se systémy a klasifikátory [USK]

vyučující:	Prof. Ing. Josef Psutka, CSc.
ročník/semestr studia:	3.ročník/LS
datum zkoušky:	X. 4. 2014
hodnocení:	1

cíl předmětu (STAG):
Cílem předmětu je seznámit studenty se základními metodami klasifikace předmětů a jevů, které jsou reprezentovány svými obrazy (vektory příznaků). Výuka bude zaměřena na klasifikátory, které jsou trénovány s podporou učitele (supervised) anebo bez učitele (unsupervised).

1.1.1 Kritérium minimální chyby.

Často nejsme schopni posoudit jednoznačně, do které třídy vektor příznaků X patří. Cílem je potom nastavit klasifikátor tak, aby ztráty způsobené chybným rozhodnutím byly minimální.

Definition 1 *Ztráta, která vznikne, jestliže obraz náležející do třídy ω_s zařadí klasifikátor do třídy ω_r: $l(\omega_r|\omega_s)$*

- předp., že obrazový prostor X obsahuje obrazy z R tříd: $\omega_1, ..., \omega_R$
- apriorní ppsti výskytu obrazů náležejících ke třídě $\omega_r => p(\omega_r), \qquad r = 1, ..., R$
- podmíněná hustota ppsti obrazu x ze třídy ω_r je $p(x|\omega_r), \qquad r = 1, ..., R$
- nechť je dána matice ztrátových funkcí:

$$l = \begin{bmatrix} l(\omega_1|\omega_1) & \dots & l(\omega_1|\omega_R) \\ \vdots & \ddots & \vdots \\ l(\omega_R|\omega_1) & \dots & l(\omega_R|\omega_R) \end{bmatrix} \tag{1.1}$$

Předpokládejme, že na vstup klasifikátoru přicházejí x pouze z ω_s a klasifikátor je bude zařazovat do ω_r podle diskriminační funkce $\omega_r = d(x, q)$.

Definition 2 *Podmíněná střední ztráta (střední ztráta podmíněná výběrem obrazů výlučně ze třídy ω_s:*

$$J(q|\omega_s) = \int_X l[d(x,q)|\omega_s] \cdot p(x|\omega_s)\, dx \tag{1.2}$$

Protože jednotlivé třídy ω_s se vyskytují s ppstí $p(\omega_s)$, bude celková střední ztráta:

$$J(q) = \sum_{s=1}^{R} J(q|\omega_s) \cdot p(\omega_s) = \int_X \sum_{s=1}^{R} l[d(x,q)|\omega_s] \cdot p(x|\omega_s) \cdot p(\omega_s)\, dx \tag{1.3}$$

Hledáme q^*, které minimalizuje $J(q)$:

$$\begin{aligned} J(q^*) = \min_q J(q) &= \int_X \min_q \sum_{s=1}^{R} l[d(x,q)|\omega_s] \cdot p(x|\omega_s) \cdot p(\omega_s)\, dx = \\ &= \int_X \min_r \sum_{r=1}^{R} l(\omega_r|\omega_s) \cdot p(x|\omega_s) \cdot p(\omega_s)\, dx = \int_X \min_r L_x(\omega_r)\, dx \end{aligned} \tag{1.4}$$

Místo minima $J(q)$ hledáme minimum $L_x(\omega_r) = \sum_{r=1}^{R} l(\omega_r|\omega_s) \cdot p(x|\omega_s) \cdot p(\omega_s), \qquad r = 1, ..., R.$

Při klasifikaci podle funkce $L_x(\omega_r)$ by se postupovalo tak, že pro daný x by se vyčíslily všechny $L_x(\omega_r), r = 1, ..., R$ a obraz x by se přiřadil do té třídy ω_s, pro kterou by byla ztráta minimální. Je zřejmé, že různou volbou ztrátové funkce $l(\omega_r|\omega_s)$ dostáváme různý tvar rozhodovacího pravidla. Předpokládejme, že ztrátová funkce je zvolena tak, že při správném rozhodnutí přiřadí ztrátu 0 a při jakémkoliv špatném rozhodnutí ztrátu 1 (penalta 0/1).

$$l(\omega_r|\omega_s) = 1 - \delta_{rs}, \qquad \delta_{rs} = \begin{cases} 1 & r = s \\ 0 & r \neq s \end{cases} \tag{1.5}$$

Po dosazení:

$$\begin{aligned} L_x(\omega_r) &= \sum_{s=1}^{R} (1 - \delta_{rs}) p(x|\omega_s) \cdot p(\omega_s) = \sum_{s=1}^{R} p(x|\omega_s) \cdot p(\omega_s) - \sum_{s=1}^{R} \delta_{rs}\, p(x|\omega_s) \cdot p(\omega_s) \\ &= \sum_{s=1}^{R} [p(x|\omega_s) \cdot p(\omega_s)] - p(x|\omega_r) \cdot p(\omega_r) \end{aligned} \tag{1.6}$$

Platí známý Bayesův vztah:

$$\boxed{p(\omega_s|x) = \frac{p(x|\omega_s) \cdot p(\omega_s)}{p(x)}} \qquad , \tag{1.7}$$

kde $p(\omega_s|x)$ je aposteriorní pravděpodobnost, která vyjadřuje ppst třídy ω_s za předpokladu, že je na vstupu klasifikátoru obraz x.

$p(x|\omega_s)$ ... ppst x za předpokladu, že patří do ω_s

$p(\omega_s)$ ... apriorní ppst třídy ω_s

$p(x)$ ... ppst obrazu x (celková hustota funkce do obrazového prostoru)

$$\sum_{s=1}^{R} p(\omega_s|x) \stackrel{!}{=} 1 = \sum_{s=1}^{R} \frac{p(x|\omega_s) \cdot p(\omega_s)}{p(x)} => p(x) = \sum_{s=1}^{R} p(x|\omega_s) \cdot p(\omega_s) \tag{1.8}$$

Dosadíme: $L_x(\omega_r) = p(x) - p(x|\omega_r) \cdot p(\omega_r)$. Hodnota $p(x)$ je pro všechny třídy konstantní a jedná se v podstatě o aditivní konstantu, takže lze definovat novou funkci $L'_x(\omega_r) = p(x|\omega_r) \cdot p(\omega_r)$. Klasifikace zde probíhá tak, že se hledá takové zařazení ω_s, pro které je $L'_x(\omega_r)$ maximální:

$$\omega_r^* = \underset{r}{\operatorname{argmax}}\, p(x|\omega_r) \cdot p(\omega_r), \qquad r = 1, ..., R \tag{1.9}$$

1.1.2 Pravděpodobnostní diskriminační funkce. Souvislost s klasifikátory podle lineární diskriminační funkce, podle nejmenší vzdálenosti, podle nejbližšího souseda a podle k-nejbližšího souseda.

Kritérium minimální chyby se často označuje jako Bayesovo kritérium. Klasifikaci lze zajistit s využitím diskriminačních funkcí:

$$g'_r(x) = p(x|\omega_r) \cdot p(\omega_r), \qquad r = 1, ..., R \tag{1.10}$$

Klasifikátor pracující podle Bayesova kritéria se nazývá Bayesův klasifikátor. Pro jeho konstrukci je třeba znát hodnoty apriorní pravděpodobnosti a hustoty pravděpodobnosti pro každou třídu. Rozhodnutí, do které třídy neznámý obraz x patří, se provede podle hodnoty $g'_r(x)$ výběrem maxima.

Velmi často se místo diskriminační funkce $g'_r(x)$ používá její přirozený logaritmus:

$$g_r(x) = ln\, g'_r(x) = ln\, p(x|\omega_r) + ln\, p(\omega_r), \qquad r = 1, ..., R \tag{1.11}$$

Např. pro $R = 3$ a jednosložkový vektor x:

Předpokládejme, že obrazy v jednotlivých třídách vyhovují normálnímu rozložení (velmi častý případ). Pro obrazy v r-té třídě nechť platí:

$$\boxed{p(x|\omega_r) = \frac{1}{(2\pi)^{\frac{n}{2}} \cdot \sqrt{det\, C_r}} \cdot e^{-\frac{1}{2} \cdot (x-\mu_r)^T \cdot C_r^{-1} \cdot (x-\mu_r)}} \tag{1.12}$$

$\mu_r = E\{x\}_{x\in\omega_r}$ ... vektor středních hodnot obrazů r-té třídy

$C_r = E\{(x-\mu_r)\cdot(x-\mu_r)^T\}_{x\in\omega_r}$ ... kovarianční matice r-té třídy

Dosadíme do diskriminační funkce:

$$g_r(x) = ln\, p(x|\omega_r) + ln\, p(\omega_r) = -\frac{n}{2}\, ln\, 2\pi - \frac{1}{2}\, ln(det\, C_r) - \frac{1}{2}\cdot(x-\mu_r)^T\cdot C_r^{-1}\cdot(x-\mu_r) + ln\, p(\omega_r) \tag{1.13}$$

Podle tvarů kovariančních matic C_r, hodnot μ_r a $p(\omega_r)$ dostáváme typické tvary diskriminačních funkcí.

1. **Obecné kovarianční matice** $C_r, r = 1, ..., R$

   Hyperplochy konstantních hodnot diskriminačních funkcí $g_r(x)$ jsou n-dimenzionální hyperelipsoidy (různě natočené a různě velké). Rozdělující hyperplocha mezi třídou ω_r a ω_s, tj. plocha, která je geometrickým místem shodných hodnot diskriminačních funkcí $g_r(x)$ a $g_s(x)$.

   $$\begin{aligned}\varphi_{rs}(x) &= g_r(x) - g_s(x) \stackrel{!}{=} 0 \\ &= -\frac{1}{2}\, ln\left[\frac{det\, C_s}{det\, C_r}\right] + ln\left[\frac{p(\omega_r)}{p(\omega_s)}\right] - \frac{1}{2}\cdot(x-\mu_r)^T\cdot C_r^{-1}\cdot(x-\mu_r) + \frac{1}{2}\cdot(x-\mu_s)^T\cdot C_s^{-1}\cdot(x-\mu_s)\end{aligned} \tag{1.14}$$

   Rozdělující hyperplochy mohou být podle tvaru C_r a C_s např. n-dimenzionální hyperroviny, hyperelipsoidy, hyperparaboloidy apod.

2. **Všechny třídy mají stejnou kovarianční matici** $C_r = C, \forall r = 1, ..., R$

   Shluky vzorků všech tříd vytvářejí stejně orientované a velké n-dimenzionální elipsoidy.

   $$g_r(x) = -\frac{1}{2}x^T C_r^{-1}x + \mu_r^T C_r^{-1}x - \frac{1}{2}\mu_r^T C_r^{-1}\mu_r + ln\, p(\omega_r) - \frac{n}{2}\, ln\, 2\pi - \frac{1}{2}\, ln(det\, C) \tag{1.15}$$

   Plochy konstantních velikostí diskriminačních funkcí jsou n-rozměrné elipsoidy, které mají stejný tvar a jsou stejně orientovány. Rozdělující plochu $\varphi_{rs}(x)$ mezi třídami ω_r a ω_s lze vyjádřit:

   $$\begin{aligned}\varphi_{rs}(x) &= (\mu_r-\mu_s)^T C^{-1}x - \frac{1}{2}\mu_r^T C^{-1}\mu_r + \frac{1}{2}\mu_s^T C^{-1}\mu_s + ln\, p(\omega_r) - ln\, p(\omega_s) = \\ &= \varphi_{rsn}x_n + \cdots + \varphi_{rs1}x_1 + \varphi_{rs0} = \varphi_{rs}^T x + \varphi_{rs0}\end{aligned} \tag{1.16}$$

   Rozdělující plocha mezi třídami ω_r a ω_s je n-dimenzionální *rovina*. Jedná se tedy o n-dimenzionální *lineární diskriminační funkci*.

3. **Všechny třídy mají stejnou diagonální kov. matici** $C_r = C = \delta^2 I, r = 1, ..., R$

   Předpokladem je, že obrazy každé třídy mají statisticky nezávislé příznaky a každý příznak má stejnou varianci δ^2. Geometricky to odpovídá situaci, kdy vzorky každé třídy vytváří shluky tvaru n-dimenzionálních koulí centrovaných kolem příslušné střední hodnoty. Potom $det\, C = \delta^{2n}$ a $C^{-1} = \frac{1}{\delta^2}I$. Předpokládejme, že všechny třídy jsou stejně pravděpodobné,

tj. $p(\omega_r) = p(\omega), \forall r = 1, ..., R$. Potom:

$$g_r(x) = -\frac{1}{2\delta^2}||x - \mu_r||^2 + ln\, p(\omega) - \frac{n}{2}\, ln\, 2\pi - \frac{1}{2}\, ln(\delta^{2n}) = -k_1 \cdot ||x - \mu_r||^2 + k_2 \quad (1.17)$$

Konstanty $k_1 > 0$ a k_2 jsou shodné pro všechny třídy a výraz $||x - \mu_r||^2$ představuje kvadrát Euklidovské vzdálenosti mezi vektorem x a střední hodnotou ω_r. Klasifikátor zařadí neznámý obraz x do té třídy ω_r, pro kterou je $g_r(x)$ maximální. Z výrazu pro $g_r(x)$ vyplývá, že $g_r(x)$ bude tím větší, čím bude $||x - \mu_r||^2$ menší ($k_1 > 0$). Jedná se tedy v podstatě o *klasifikátor podle minimální vzdálenosti.*

Klasifikace podle minimální vzdálenosti

Diskriminační funkce: $g_r^*(x) = ||x - \mu_r||^2$. Klasifikátor zařadí neznámý obraz x do té třídy, pro kterou bude $g_r^*(x)$ minimální ($\omega_r^* = \min_r d^2(x, \mu_r)$). Není to určitě nejlepší klasifikátor, ale je tu velká lákavost ho používat, protože stačí jediný obraz na třídu. Rozdělující nadrovina má tvar:

$$\begin{aligned} \varphi_{rs}(x) &= -k1 \cdot ||x - \mu_r||^2 + k_2 - \left[-k_1 \cdot ||x - \mu_s||^2 + k_2\right] = \\ &= k_1 \cdot \left[x^T x - 2\mu_s x + \mu_s^T \mu_s - x^T x - 2\mu_r x + \mu_r^T \mu_r\right] \stackrel{!}{=} 0 \\ &=> (\mu_r - \mu_s)^T x - \frac{1}{2}(\mu_r^T \mu_r - \mu_s^T \mu_s) = 0 \end{aligned} \quad (1.18)$$

Rozdělující nadplochy mezi třídami jsou *lineární*, jsou to n-dimenzionální roviny kolmé na úsečku $\mu_r - \mu_s$, kterou půlí [1]. Tento klasifikátor je velmi jednoduchý na implementaci - pro jeho nastavení stačí získat střední hodnoty každé třídy a pro neznámý obraz x ve fázi klasifikace vypočítat vzdálenost ke všem středním hodnotám (též nazýván *klasifikátor se vzorovými etalony.* Pro svou jednoduchost je často nasazován i v případech, kdy není zabezpečena jeho optimální funkce podle kritéria minimální chyby (např. je málo početná trénovací množina nebo není znám typ rozložení nebo není známa disperzní matice ap.). To vede k negativním vlivům klasifikátoru:

- vzhledem k tomu, že využívá pouze střední hodnoty, nerespektuje tvar shluků jednotlivých tříd (pokud je odlišný od $C_r = C = \delta^2 I$; tvar shluku koresponduje s tvarem disperzní matice).
- nerespektuje případné odlišné apriorní pravděpodobnosti jednotlivých tříd

Dobrých výsledků dosáhneme, když budou třídy dobře distribuované (střední hodnoty dostatečně vzdálené, shluky dostatečně kompaktní a jednotlivé třídy stejně pravděpodobné).

Klasifikace podle nejbližšího souseda (Nearest Neighbour Classifier)

Uvedené nevýhody klasifikátoru podle minimální vzdálenosti lze zmírnit často tím, že využijeme více vzorových etalonů pro každou třídu. Zvolíme-li pro každou třídu ω_r S_r vzorových etalonů:

[1] Klasifikátor podle minimální vzdálenosti je ekvivalentní co do struktury lineárnímu klasifikátoru s R diskriminačními funkcemi, který může vytvořit až $\frac{R(R-1)}{2}$ rozdělujících nadrovin. Má však obecně jiné parametry, tj. klasifikuje obecně jiným způsobem než lineární klasifikátor.

$\mu_{r1}, \mu_{r2}, \ldots, \mu_{rS_r}$, pak klasifikace probíhá podle pravidla vyjádřeného vztahem:

$$\omega_r^* = \underset{s,r}{\operatorname{argmin}} ||x - \mu rs|| = \underset{s,r}{\operatorname{argmin}}\, d(x, \mu rs) \tag{1.19}$$

Obraz x se tedy zařadí do té třídy ω_r, jejíž některý etalon má mezi všemi ostatními etalony nejmenší vzdálenost od x. Tento způsob klasifikace má tu výhodu, že při dostatečně rozsáhlé trénovací množině se tvar rozdělujících funkcí pro jednotlivé třídy "blíží"Bayesovskému klasifikátoru. Na druhou stranu to však znamená značné zvýšení výpočetních nároků (klasifikátor si musí neustále pamatovat celou množinu vzorových etalonů - celou trénovací množinu) a při klasifikaci musíme počítat vzdálenost neznámého obrazu x ke všem vzorům.

Klasifikace podle k-nejbližších sousedů (k-Nearest Neighbour Classifier)

Lepších výsledků lze často dosáhnout využitím tzv. rozhodovacího pravidla, kdy nejprve vyčíslíme všechny vzdálenosti $||x-\mu_{rs}||\forall r = 1, ..., R\forall s = 1, ..., S_r$ a pak je pro každou třídu ω_r uspořádáme tak, aby pro nový soubor $||x - \mu_{r[s]}||$ platilo:

$$||x - \mu_{r[1]}|| \leq ||x - \mu_{r[2]}|| \leq \cdots \leq ||x - \mu_{r[S_r]}||$$

Klasifikátor pak zařadí obraz x do třídy ω_r^* podle minima průměrné vzdálenosti k-nejbližších sousedů:

$$\omega_r^* = \underset{r}{\operatorname{argmin}} \frac{1}{k}\sum_{k=1}^{k}||x - \mu_{r[i]}||, \qquad r = 1, ..., R \tag{1.20}$$

Nevýhody:

- musím si pamatovat všechny obrazy
- při každé klasifikaci náročné výpočty

1.1.3 Klasifikátor s lineární diskriminační funkcí. Klasifikace do dvou a do více tříd.

Pokud obrazy v jednotlivých třídách podléhají normálnímu rozložení a všechny třídy vykazují stejnou kovarianční matici C, je optimální nastavení klasifikátoru podle kritéria minimální chyby (Bayesova kritéria) zabezpečeno *lineárními diskriminačními funkcemi*. Vzhledem k jejich výhodným analytickým vlastnostem se jich ovšem využívá i v případech, kdy výše uvedené podmínky splněny nejsou a nebo, a to je častější případ, kdy ověření platnosti těchto podmínek je nepřiměřeně náročné (např. nelze statisticky prokázat typ rozložení vzhledem k malému počtu obrazů ap.). Zvolíme-li v takovém případě rozhodovací pravidlo založené na lineárních diskriminačních funkcích, musíme mít vždy na paměti, že jsme nezvolili optimální řešení s hlediska Bayesova kriteria minimální chyby. Přesto je třeba říci, že v případech, kdy obrazy jednotlivých tříd jsou dobře distribuované, tj. vytvářejí kompaktní shluky, které jsou od sebe dostatečně vzdálené (lineárně separabilní třídy) toto zjednodušení dostatečně vyhovuje.

Uvažme lineární diskriminační funkci $g(x) = q_0 + q_1x_1 + \ldots q_nx_n = q_0 + \sum_{i=1}^{n} q_ix_i$, kde q_i jsou váhy funkce a q_0 je práh funkce.

Dále mějme $||q|| = \sqrt{q_1^2 + q_2^2 + \cdots + q_n^2}$. Pro $n = 2$:

Klasifikace do dvou tříd (dichotomie)

Při klasifikaci do dvou tříd ω_1 a ω_2 stačí k rozhodnutí jediná diskriminační funkce:

$$g(x) = q_0 + \sum_{i=1}^{n} q_i x_i \tag{1.21}$$

Pro $g(x) > 0$ je $x \in \omega_1$, pro $g(x) < 0$ je $x \in \omega_2$.

Klasifikace do více tříd

(a) Předpokládejme, že obrazy každé třídy jsou *lineárně separovatelné* od obrazů všech ostatních tříd. Pak diskriminační funkce mezi třídami ω_r a $\bar{\omega_r}$ je:

$$g_r(x) = q_{r,0} + \sum_{i=1}^{n} q_{r,i} x_i \tag{1.22}$$

a platí, že pro $x \in \omega_r$ je $g_r(x) > 0$ a pro $x \in \bar{\omega_r}$ je $g_r(x) < 0$. Klasifikátor pak rozhodne o zařazení x do té třídy $\omega_r (r = 1, ..., R)$ pro níž je diskriminační funkce $g_r(x) > 0$. Problém je však v tom, že se může stát, že pro neznámé x bude hodnota více než jedné diskriminační funkce větší než 0. V takovém případě klasifikátor není schopen rozhodnout [2].

Př. ($n = 2$, $R = 3$):

(b) Předpokládejme, že obrazy každé třídy jsou *po dvojicích lineárně separovatelné* od všech ostatních tříd. V tomto případě existuje celkově $\frac{R(R-1)}{2}$ diskriminačních funkcí $\varphi_{rs}(x); r, s =$

[2] Tento způsob klasifikace má jistá omezení, např. v prostoru dimenze $n = 2$ lze takto rozdělit maximálně $R = 3$ dobře distribuované třídy.

$1, ..., R \wedge r \neq s$, které vytvářejí rozdělující roviny mezi obrazy všech dvojic tříd. Pro obraz $x \in \omega_r$ pak platí $\varphi_{rs}(x) > 0 \, \forall s \neq r$, viz[3].

Př. ($n = 2$, $R = 4$):

(c) Předpokládejme[4], že existují diskriminační funkce $g_r(x), r = 1, ..., R$ z případu ad a). Vytvoříme rozdělující hyperplochy mezi třídami r a s.

$$\varphi_{rs}(x) = g_r(x) - g_s(x) \stackrel{!}{=} 0 \tag{1.23}$$

Pro $\varphi_{rs}(x) > 0$ je $g_r(x) > g_s(x)$. Z toho vyplývá, že klasifikátor zařadí x do ω_r, jestliže $g_r(x) > g_s(x) \, \forall s = 1, ..., R$; $s \neq r$. Viz poznámky[5].

Hodnocení: Je zřejmé, že případ ad a) není vhodný k aplikování vzhledem k vytváření rozsáhlých oblastí, ve kterých nejsme schopni provést jednoznačné přiřazení. Rozhodnutí mezi případem ad b) a ad c) závisí do značné míry na intuici (zvláště v prostorech vyšší dimenze). Obecně lze říci, že případ ad b) vyžaduje určení $\frac{R(R-1)}{2}$ diskriminačních funkcí $\varphi_{rs}(x)$, kdežto případ ad c) požaduje nalezení pouze R diskriminačních funkcí $g_r(x)$. Jestliže se však počet tříd R blíží dimenzi n obrazového prostoru nebo se očekává, že obrazy jednotlivých tříd jsou špatně distribuované, bude možná postup podle ad b) lepším řešením.

1.1.4 Metody nastavování klasifikátorů (trénování klasifikátorů).

(a) Známe-li celou trénovací množinu apriori, můžeme se pokusit o *analytické řešení:*

- *pravděpodobnostní diskriminační funkce*: je třeba určit a prokázat typ rozložení a hodnoty parametrů rozložení včetně apriorní pravděpodobnosti tříd

[3] Samozřejmě platí $\varphi_{rs}(x) = -\varphi_{sr}(x)$. V mnoha případech se nevyužívá všech $\frac{R(R-1)}{2}$ diskriminačních funkcí. V tomto případě se opět objevují oblasti, pro které nejsme schopni rozhodnout a zařazení x.

[4] Vylepšení ad a).

[5] Rozdělující funkce $\varphi_{rs}(x)$ rozdělují obrazový prostor bezezbytku (nejsou hluché oblasti, kde není možno provést přiřazení).

- *klasifikátor dle minima vzdálenosti*: je třeba určit střední hodnotu obrazů v každé třídě
- *klasifikátor podle nejbližšího či k-nejbližšího souseda*

Pro prostory vyšší dimenze nepoužitelné, navíc je třeba si pamatovat celou trénovací množinu. Dále není umožněno dotrénování klasifikátoru, pokud se objeví nové informace o trénovací množině. Častou chybou je nerespektování apriorních pravděpodobností tříd.

(b) Trénovací množinu neznáme apriori: *metody učení* Učící se klasifikátor má dvě fáze:

- *fáze učení*: postupně předkládány dvojice $[x_k, \Omega_k]$ z trénovací množiny, nastavujeme parametry q tak, aby pro $k \to \infty$ bude $q \to q^*$ (optimální nastavení, minimální střední hodnota ztrát).
- *fáze klasifikace*: využívá se zkušenosti nashromážděné v parametrech q k predikci neznámých obrazů. Klasifikátor se chová jako jednoúčelový automat.

Střední ztráta:

$$J(q) = \int_{X\times O} Q(x, \Omega, q)\, dF(x, \Omega) = \sum_{r=1}^{R} p(\omega_r) \int_X Q(x, \Omega, q) p(x|\omega_r)\, dx \tag{1.24}$$

Úkolem je najít takové q^*, které minimalizuje $J(q)$.

$$\operatorname*{grad}_q J(q^*) = \int_{X\times O} \operatorname*{grad}_q Q(x, \Omega, q^*)\, dF(x, \Omega) \stackrel{!}{=} 0 \tag{1.25}$$

Běžně ovšem neznáme distribuční (ani hustotní) funkce, proto se obracíme na rekurentní algoritmy, které obcházejí přímé řešení rovnice. Existují dva přístupy:

- metody učení založené na odhadování hustot ppsti
- metody učení založené na přímé minimalizaci ztrát

Metody učení založené na odhadování hustot ppsti

Snaha stanovit distribuční funkci $dF(x, \Omega)$ z trénovací množiny, poté se využije kritérium minimální chyby a dostáváme soustavu diskriminačních funkcí $g_r(x) = p(x|\omega_r) \cdot p(\omega_r)$. Hledáme odhady $\hat{p}(x|\omega_r)$ a $\hat{p}(\omega_r)$. Žádané vlastnosti:

- *nestrannost*: má zaručovat, že se odhad bude v průměru pohybovat kolem neznámé odhadované veličiny
- *konzistence*: s rostoucím počtem obrazů trénovací množiny se odhad blíží stále více k odhadované veličině
- *eficience*: eficientní odhad je odhad s nejmenší disperzí

Je-li trénovací množina vybrána nezávisle, lze odhad $\hat{p}(\omega_r)$ apriorní ppsti určit podle:

$$\hat{p}(\omega_r) = \frac{K_r}{K}, \qquad r = 1, ..., R \tag{1.26}$$

kde K_r je počet obrazů trénovací množiny, které patří do třídy ω_r a K je celkový počet obrazů v trénovací množině. Podle velikosti apriorní informace o hledané hustotě rozdělujeme metody získávání odhadů hustotní funkce na *parametrické* a *neparametrické*.

1. *Parametrické metody*: známe informaci o tvaru hustotní funkce $p(x|\omega_r)$, ale neznáme parametry q, který rozložení blíže popisuje (např. u Gausse: μ_r a δ_r).

   - *Metoda momentů*

     Teoretické momenty náhodných veličin se porovnávají s výběrovými momenty do takového stupně l, kolik je neznámých parametrů.

     výběrový průměr: $\bar{x} = \frac{1}{K}\sum_{k=1}^{K} x_k$

     výběrový rozptyl: $S = \frac{1}{K-1}\sum_{k=1}^{K}(x_k - \bar{x})(x_k - \bar{x})^T$

     l-tý výběrový moment: $M_l = \frac{1}{K}\sum_{k=1}^{K} x_k^l$

   - *Metoda nejlepších nestranných lineárních odhadů*

     Odhad parametrů q se uvažuje jako lineární funkce obrazů $x_1, ..., x_K$: $\hat{q} = c_1 x_1 + ... + c_K x_K$. Koeficienty $c_k, k = 1, ..., K$ se určují z podmínek nestrannosti a minimální disperze odhadu $\hat{q}$. Platí $E\,\hat{q} = q$ a odhad $\hat{q}$ parametru q je eficientní, jestliže minimalizuje stopu disperzní matice $tr\,D\,\hat{q}$.

   - *Metoda maximální věrohodnosti*

     Metoda je založena na maximalizaci tzv. Fisherovy funkce věrohodnosti:

     $$L(x_1, ..., x_K|q) = \prod_{k=1}^{K} p(x_k|q) \tag{1.27}$$

     Hledáme takový parametr q, pro který je funkce maximální. Místo tohoto maxima lze hledat maximum logaritmu této věrohodnostní funkce $\operatorname{grad}_q ln\, L(x_1, ..., x_K|q) \stackrel{!}{=} 0$. Např. pro normální rozdělení je nejlepším odhadem střední hodnoty aritmetický průměr $\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i$ a nejlepším odhadem kovarianční matice je:

     $$C = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)(x_i - \mu)^T \tag{1.28}$$

2. *Neparametrické metody*: máme nulovou apriorní informaci, musíme odhadovat celý tvar hustotní funkce.

   - *Metoda histogramu*

Obrazový prostor rozdělíme na L disjunktních podmnožin $A_l, l = 1, ..., L$ (obvykle to jsou n-rozměrné intervaly). Symbolem c_l označíme počet obrazů x_k, které padnou do A_l, dělený číslem K. Za odhad $\hat{p}(x|\omega_r)$ pak bereme po částech konstantní funkci, která na intervalu A_l nabývá c_l. Nevýhodou je nutnost znalosti apriorního rozdělení obrazového prostoru na intervaly před fází učení. Odhad hustotní funkce:

$$\hat{p}(x|\omega_r) = \sum_{l=1}^{L} c_l \cdot \varphi_l(x), \qquad \varphi_l = \begin{cases} 1 & x \in A_l \\ 0 & jinak \end{cases} \tag{1.29}$$

Metody učení založené na přímé minimalizaci ztrát

Cílem je navrhnout parametry klasifikátoru, které budou minimalizovat ztrátu, rekurentním vypočítáváním odhadů $\hat{q}(0) \to \hat{q}(1) \to \cdots \to \hat{q}^*$.

$$\operatorname*{grad}_q J(q^*) = \int_{X \times O} \operatorname*{grad}_q Q(x, \Omega, q^*)\, dF(x, \Omega) = 0 \tag{1.30}$$

Nabízí se využití gradientních metod. V praktických úlohách neznáme distribuční funkci a nelze tak vyčíslit grad $J(q)$. Navíc, při pevném q hodnota funkce $Q(x, \Omega, q^*)$ náhodně kolísá v závislosti na x a Ω - to je u gradientních metod nepoužitelné. Řešení rovnice hledáme pomocí metody *stochastických aproximací*. Základní algoritmus přímé minimalizace ztrát:

$$q(k+1) = q(k) - C_{k+1} \operatorname*{grad}_q Q[x(k+1), \Omega(k+1), q(k)] \qquad k = 1, ..., K \tag{1.31}$$

C_k je čtvercová matice (obvykle $C_k = c_k \cdot I$), $[x(k), \Omega(k)]$ je k-tá dvojice trénovací množiny a q je vektor parametrů. Vhodnou volbou $Q(x, \Omega, q)$ a C_k lze získat téměř všechny algoritmy přímé minimalizace ztrát.

Položíme $x_0 = 1$: $x^T = [x_0, x_1, ..., x_n]$ a váhový vektor rozšíříme o práh q_0: $q^T = [q_0, q_1, ..., q_n]$. Diskriminační funkce má tvar $g(x) = q^T \cdot x$ a rozhodovací pravidlo $\omega = \operatorname{sign} g(x)$. Po zavedení pásma necitlivosti δ volíme:

$$\operatorname*{grad}_q Q(x, \Omega, q) = \begin{cases} 0 & q^T x \Omega \geq \delta \\ -x\Omega & q^T x \Omega < \delta \end{cases} \tag{1.32}$$

Dosadíme-li do vztahu pro rekurentní výpočet q, dostaneme algoritmus učení:

$$\begin{aligned} q(k+1) &= q(k) - C_{k+1} \operatorname*{grad}_q Q[x(k+1), \Omega(k+1), q(k)] = \\ &= \begin{cases} q(k) & q^T(k)x(k+1)\Omega(k+1) \geq 0 \\ q(k) + C_{k+1}x(k+1)\Omega(k+1) & q^T(k)x(k+1)\Omega(k+1) < 0 \end{cases} \end{aligned} \tag{1.33}$$

(a) *Rosenblattův algoritmus*

$C_k = C_{k+1} = 1$, $\delta = 0$:

$$q(k+1) = \begin{cases} q(k) & q^T(k)x(k+1)\Omega(k+1) \geq 0 \\ q(k) + x(k+1)\Omega(k+1) & q^T(k)x(k+1)\Omega(k+1) < 0 \end{cases} \tag{1.34}$$

Nedostatkem je, že nikdy nezjistíme absolutně přesný klasifikátor.

(b) *Metoda konstantních přírůstků*

$C_k = \frac{\beta}{||x(k)||^2}$, $\beta > 0$

$$q(k+1) = \begin{cases} q(k) & q^T(k)x(k+1)\Omega(k+1) \geq \delta \\ q(k) + \frac{\beta}{||x(k+1)||^2}x(k+1)\Omega(k+1) & q^T(k)x(k+1)\Omega(k+1) < \delta \end{cases} \quad (1.35)$$

(c) *Upravená metoda konstantních přírůstků*

Připočítáváme vektor $\frac{\beta}{||x(k+1)||^2}x(k+1)\Omega(k+1)$ tolikrát, až pro určitý obraz $x(k+1)$ dosáhneme správné klasifikace, tj. bude platit $q^T(k+1)x(k+1)\Omega(k+1) \geq \delta$. Může se stát, že nikdy nedojdeme k řešení.

(d) *Relaxační metoda učení*

$$q(k+1) = \begin{cases} q(k) & q^T(k)x(k+1)\Omega(k+1) \geq \delta \\ q(k) + 2C_{k+1}x(k+1)\Omega(k+1)[\delta - q^T(k)x(k+1)\Omega(k+1)] & q^T(k)x(k+1)\Omega(k+1) < \delta \end{cases} \quad (1.36)$$

kde $c_k = \frac{\sigma}{2||x(k)||^2}$ a $\sigma \in (0, 2)$.

1.1.5 Metody shlukové analýzy (učení bez učitele).

Někdy se stane, že je k dispozici jen trénovací množina bez údajů o správné klasifikaci, někdy chybí i informace o počtu tříd. Úkolem je nalézt shluky obrazů, tj. takové skupiny jejichž prvky jsou si vzájemně podobné (geometricky blízké). Lze aplikovat pouze na data, kde ty shluky opravdu jsou.

Požadavky pro míry podobnosti: $d(x_i, x_i) = 0$; $d(x_i, x_j) \geq 0$, $i \neq j$; $d(x_i, x_j) = d(x_j, x_i)$.

Míry podobnosti mezi dvěma obrazy x a x':

- $d(x, x') = ||x - x'||$
- Euklidova vzdálenost: $d(x, x') = \sqrt{\sum_{i=1}^{n}(x_i - x'_i)^2}$

Míry podobnosti mezi dvěma shluky T_i a T_j:

- minimální míra: $D_{min}(T_i, T_j) = \min_{x \in T_i; x' \in T_j} d(x, x')$
- maximální míra: $D_{max}(T_i, T_j) = \max_{x \in T_i; x' \in T_j} d(x, x')$
- průměrná míra: $D_{mean}(T_i, T_j) = \frac{1}{s_i \cdot s_j} \sum_{x \in T_i} \sum_{x' \in T_j} d(x, x')$

- centroidní míra: $D_c(T_i, T_j) = ||\frac{1}{s_i}\sum_{x\in T_i} x - \frac{1}{s_j}\sum_{x'\in T_j} x'|| = d(\bar{x}, \bar{x}')$

Nehierarchické shlukovací metody

Snaha provést optimální rozklad dané množiny T do předem známého počtu R shluků. Hledáme takové rozdělení, pro které nabývá kritérium J extrém.

$$J = \sum_{i=1}^{R} J_i = \sum_{i=1}^{R}\sum_{x\in T_i} d^2(x, \mu_i) \tag{1.37}$$

1. **McQueenova metoda k-means (1967)** Algoritmus se snaží minimalizovat celkový ukazatel jakosti J rozdělením obrazů do shluků.

   Dáno:

   $T = \{x_i\}_{i=1}^{N}$ ... obrazy trénovací množiny

   R ... počet tříd

   $T_i(k)$ ... množina obrazů i-tého shluku v k-tém kroku algoritmu

   $\mu_i(k)$ ... střední (průměrná) hodnota i-tého shluku v k-tém kroku

   $J_i(k)$ ... hodnota kritéria i-tého shluku v k-tém kroku

   $s_i(k)$ ... počet obrazů ve shluku T_i v k-tém kroku

   Postup:

   (i) Zvol R počátečních středů shluků $\mu_1(1), ..., \mu_R(1)$.

   (ii) V k-tém iteračním kroku se rozdělují obrazy trénovací množiny do R shluků $T_1(k), ..., T_R(k)$ podle vztahu $x \in T_j(k)$ jestliže $d(x, \mu_j(k)) < d(x, \mu_i(k))\, \forall i, j = 1, ...R;\ i \neq j$. Vztah je postupně aplikován pro všechny obrazy z množiny T.

   (iii) Z výsledků předchozího kroku vypočti pro každý shluk nový střed, tj. $\mu_j(k+1)$, $j = 1, ..., R$ klasickým zprůměrováním (zaručí, že $J_j(k+1$ bude minimální):

   $$\mu_j(k+1) = \frac{1}{s_j(k)}\sum_{x\in T_j(k)} x, \qquad j = 1, ..., R \tag{1.38}$$

   (iv) Jestliže $\mu_j(k+1) = \mu_j(k)\, j = 1, .., R$, algoritmus dokonvergoval a procedura je ukončena. Jinak jdi na krok (ii). Alternativně lze proces ukončit v případě, že pokles hodnoty kriteriální funkce je již nevýznamný.

   Funkce metody je ovlivněna zejména specifickým počtem shluků a volbou počátečních středů shluků. Metoda sice pro vhodná data poskytuje přijatelné výsledky, ale dosažení globálního minima ukazatele jakosti procesu shlukování není zaručeno (konvergence nejčastěji končí v nějakém lokálním minimu).

2. **Iterativní optimalizace** V k-tém iteračním kroku změníme zařazení jednoho obrazu.

Postup:

(i) Proveď počáteční rozklad N obrazů do R shluků a vypočti hodnotu kriteriální funkce J a urči $\mu_i(1)$, $i = 1, ..., R$.

(ii) Vyber obraz $\hat{x}$, kde např. $\hat{x} \in T_i$ (nejlépe nějaký systematický výběr).

(iii) Jestliže $s_i(k) = 1$, jdi na bod (vi)

(iv) Vypočti:

$$\begin{aligned} A_i &= \frac{s_i(k)}{s_i(k)-1} d^2(\hat{x}, \mu_i(k)) \\ A_j &= \frac{s_j(k)}{s_j(k)+1} d^2(\hat{x}, \mu_j(k)), \qquad \forall j \neq i \end{aligned} \tag{1.39}$$

Urči $j^* = \underset{j}{\operatorname{argmin}} A_j$. Jestliže $A_i > A_{j^*}$, přesuň $\hat{x}$ do T_{j^*}. Jinak ponech $\hat{x}$ v T_i.

(v) Aktualizuj J, μ_i a μ_{j^*}.

(vi) Jestliže se J po pokusu přesunout postupně všech N obrazů trénovací množiny nezměnila, proces ukonči. Jinak přejdi do bodu (ii).

Pokud bychom chtěli dosáhnout opravdu globálního minima J, mohli bychom vyzkoušet teoreticky všechny možnosti rozkladu, ale existuje obecně $\frac{R^N}{R!}$ možností (pro $R = 3$ a $N = 100$ je to $\approx 10^{47}$ možností).

Hierarchické shlukovací metody

Většinou aplikujeme, pokud není známa informace o počtu tříd.

(A) *Aglomerativní přístup* Vycházíme z jednotlivých obrazů, které postupně shlukujeme, až dojde ke spojení všech obrazů do jedné množiny.

1. **Metoda shlukové hladiny** Celé množině obrazů $T = \{x_1, ..., x_N\}$ postupně přiřazujeme posloupnost rozkladů B_0 až B_{N-1} a každému vzniklému shluku A přiřazujeme tzv. shlukovací hladinu $h(A) \geq 0$.

   Postup:

   (i) Rozklad B_0 tvoří jednotlivé obrazy, tj. $A_{0j} = \{x_j\}$ a $h(A_{0j}) = 0$, $j = 1, ..., N$.

   (ii) V i-tém kroku pro $1 \leq i \leq N-1$ vybereme jedinou dvojici shluků (obrazů) $A_{i-1,u}$ a $A_{i-1,v}$, pro kterou nastala nejmenší vzdálenost (ve smyslu zvolené metriky).

   - provedeme sjednocení shluků $A_{i-1,u} \cup A_{i-1,v} < A_{i,l}$, $l = 1, ..., N-1$;
   - utvoříme rozklad $B_i = \{A_{i,1}, ..., A_{i,N-1}\}$, kde všechny shluky s výjimkou sjednoceného $A_{i,l}$ přechází do B_i beze změny;
   - položíme $A_{i,l} = \min_{u,v} D(A_{i-1,u}, A_{i-1,v})$

   (iii) Bod (ii) opakujeme, až v posledním kroku procedury vznikne jediný shluk.

(B) *Divisní přístup* Celkový shluk obrazů určený ke klasifikaci postupně rozdělujeme a získáme hierarchický systém podmnožin.

1. **Jednoprůchodový heuristický algoritmus hledání shluků** Je zadána trénovací množina $T = \{x_1, ..., x_N\}$. Definujeme ve vztahu se zvolenou mírou podobnosti neznámý práh $t > 0$.

   (i) Z obrazů trénovací množiny vybereme jeden, např. $x_{(1)} \in T$ a ztotožníme ho se středem prvního shluku $\mu_1 = x_{(1)}$.

   (ii) Z trénovací množiny vybereme obraz $x_{(2)} \in T$ a vypočteme vzdálenost $d_{21} = d(x_{(2)}, \mu_1)$. Jestliže $d_{21} > t$, pak zavedeme nový shluk se středem $\mu_2 \coloneqq x_{(2)}$. Jinak přidělíme obraz $x_{(2)}$ do shluku se středem μ_1.

   (iii) Předpokládáme, že shluk se středem μ_2 byl zaveden, poté vybereme obraz $x_{(3)} \in T$ a vyčíslíme $d_{31} = d(x_{(3)}, \mu_1)$ a $d_{32} = d(x_{(3)}, \mu_2)$. Jestliže $d_{31} > t$ a $d_{32} > t$, zavedeme nový shluk se středem $\mu_3 \coloneqq x_{(3)}$. Jinak přidělíme obraz $x_{(3)}$ do shluku, k jehož středu má nejmenší vzdálenost.

   (iv) Postupujeme analogicky, až rozdělíme všechny obrazy trénovací množiny.

   Výsledky závisí na prvním vybraném středu shluku; pořadí, v němž jsou obrazy uvažovány; hodnotě t; geometrických vlastnostech dat. Algoritmus je velmi jednoduchý a rychlý a stanovuje první náhled na trénovací množinu obrazů. Podle velikosti prahu t se mění i výsledný počet shluků. Metoda vyžaduje pouze jediný průchod trénovací množinou, ale v praxi je potřeba rozsáhlé experimentování s hodnotou prahu a startovacím obrazem.

2. **Metoda řetězové mapy** Základním krokem při vytváření řetězové mapy je přeuspořádání dat. Z trénovací množiny vybereme libovolný "startovací"obraz $x_{(1)}$, najdeme jeho nejbližšího souseda $x_{(1)[1]}$ a položíme $x_{(2)} = x_{(1)[1]}$ (druhá položka seznamu). Další položkou bude nejbližší soused k $x_{(2)}$, atd. Nejbližšího souseda vždy vybíráme z množiny dosud neuspořádaných obrazů. Proces pokračuje, dokud nezískáme posloupnost ze všech obrazů trénovací množiny.

   $$\tilde{T} = \{x_{(1)}, ..., x_{(N)}\} \tag{1.40}$$

   S i-tým členem této posloupnosti potom spojíme vzdálenost $d_i = d(x_{(i),x_{i+1}}$, $i = 1, ..., N-1$. Řetězovou mapu získáme jako závislost $d_i = f(i)$. Znázorníme-li tuto závislost graficky, pak pokud jsou obrazy trénovací množiny dobře distribuovány, tj. vytvářejí v prostoru kompaktní shluky, které jsou od sebe dostatečně vzdálené, lze typicky na diagramu nalézt souvislé oblasti relativně malých hodnot d_i, které jsou od sebe odděleny (relativně) vyšší hodnotou d_j. Tyto zvýšené hodnoty d_j odpovídají hranicím shluků.

3. **Metoda MAXIMIN (maximum-minimum)** Jedná se o jednoduchý heuristický algoritmus, který je vhodný pro odhad počtu shluků v obrazovém prostoru. Je dána množina obrazů $T = \{x_k\}_{k=1}^N$ a volitelná konstanta $q > 0$.

   Postup:

   (i) Z trénovací množiny vybereme "startovací"obraz a ztotožníme ho se středem prvního shluku, tj. $\mu_1 \coloneqq x_{(1)}$.

(ii) Dále zvolíme (ve smyslu zvolené metriky) nejvzdálenější obraz k μ_1 a ztotožníme ho se středem druhého shluku, tj. $\mu_2 \coloneqq x_{(2)}$.

(iii) Pro každý obraz z trénovací množiny (bez $x_{(1)}$ a $x_{(2)}$) vyčíslíme vzdálenosti k μ_1 a k μ_2 a vždy vybereme tu minimální.

(iv) Z uchovaných minimálních vzdáleností vybereme tu maximální a porovnáme její velikost s velikostí vzdálenosti $d(\mu_1, \mu_2)$ a obraz, pro který tato maximální vzdálenost nastala, označíme jako $x_{(3)}$.

(v) Jestliže je ta maximální vzdálenost větší než $q \cdot d(\mu_1, \mu_2)$, zavedeme nový shluk se středem $\mu_3 \coloneqq x_{(3)}$, jinak shlukovací proces končí.

(vi) V dalším kroku algoritmu postupujeme stejně jako v kroku (iii), ale pro tři středy shluků. To znamená, že z minimálních vzdáleností obrazů trénovací množiny $(T \setminus \{x_{(1)}, x_{(2)}, x_{(3)}\})$ ke středům shluků μ_1, μ_2 a μ_3 vybereme maximální a opět posoudíme, je-li větší než q-násobek např. průměru vzdáleností mezi již vytvořenými středy shluků. Postup opakujeme pro vzrůstající počet (středů) shluků, dokud se nová maximální vzdálenost nedostane do sporu s podmínkou vytvoření nového středu shluku.

Po nalezení počtu shluků a jejich středů lze provést rozdělení obrazů trénovací množiny do jednotlivých shluků podle kritéria minimální vzdálenosti, přičemž za vzorové obrazy považujeme středy shluků.

4. **Metody binárního dělení** Předpokládejme, že jsme v procesu shlukování získali R shluků. V dalším postupu obvykle provádíme zařazování neznámých obrazů do nejvhodnějších shluků podle pravidla nejbližšího souseda → musíme nalézt takový shluk, jehož centroid má k neznámému obrazu x_i nejmenší vzdálenost. Při vyčerpávajícím porovnání se všemi centroidy narážíme při velkém R na velké množství operací. Metoda binárního dělení umožňuje redukci množství operací.

   - **Rovnoměrné binární dělení** Požadovaný počet výsledných shluků R musí být mocninou 2. V prvním kroku dělení je prvotní shluk obrazů trénovací množiny rozdělen na dva subshluky a každý z těchto subshluků je poté v dalším kroku rozdělen na další dva subshluky, atd. Proces končí, až je dosažen požadovaný konečný počet shluků R. Platí $R = 2^B$, kde B je počet kroků dělení. Proces lze znázornit prohledávacím stromem.

     Při rovnoměrném dělení se mechanicky rozdělují všechny koncové shluky (reprezentované v prohledávacím stromu aktuálními koncovými uzly). Při tomto procesu se může stát, že v některém subshluku zůstane několik nebo dokonce jen jediný obraz → další dělení takových shluků výrazně nepřispívá k minimalizaci celkového zkreslení.

   - **Nerovnoměrné binární dělení** Zde výsledný počet shluků R nemusí být mocninou 2. Abychom zajistili nižší celkové zkreslení při daném počtu shluků, je vhodné nedělit prohledávací strom rovnoměrně, ale v každém kroku subdělícího procesu vyčíslit zkreslení všech koncových shluků a rozdělit ten shluk, který přispívá do celkového zkreslení největší měrou. Po každém kroku algoritmu, který měl za následek rozdělení nějakého subshluku zjistíme, zda již nebylo dosaženo stanoveného počtu shluků nebo zda velkost celkového zkreslení nepoklesla pod

stanovenou mez. Pokud některá z těchto podmínek byla splněna, proces dělení končí.

1.1.6 Výběr informativních příznaků.

Velikost vektorů příznaků všech vzorků uvažovaných pro stejnou úlohu musí být shodná. Pokud není, lze přemýšlet o úpravě těchto vektorů, která obecně velmi závisí na fyzikální podstatě jednotlivých jevů (příznaků). Pokud to fyzikální podstata dovoluje, můžeme řetězce lineárně natáhnout, popř. lineárně smrštit, na požadovanou velikost. Plyne-li z fyzikální podstaty vektorů příznaků možnost lineárního kolísání (nejčastěji v časové ose), lze pro výpočet vzdálenosti takových vektorů využití metody tzv. nelineárního borcení jednoho z pozorovaných obrazů (DTW - *dynamic time warping*).

Obecně při výběru informativních příznaků je třeba zachovat minimální hodnoty disperzí uvnitř jednotlivých tříd a maximální hodnoty disperzí mezi třídami.

Extrakce

Metoda extrakce výběru minimálního počtu informativních příznaků je založena na nejlepší aproximaci původních obrazů z prostoru H_m o dimenzi m obrazy z prostoru H_n o dimenzi $n < m$, a to ve smyslu minima střední kvadratické odchylky (Karhunen-Loevův rozvoj).

Selekce

Definujeme míru diskriminativnosti množiny příznaků $J = \mathrm{tr}(\hat{T}^{-1}_{AKVI} \cdot \hat{T}_{AKT})$. Ta se rovná největšímu vlastnímu číslu matice $T^{-1}_{AKVI} \cdot T_{AKT}$. Výběr příznaků založený na míře diskriminativnosti se provádí iterativně:

(i) Vypočte se míra diskriminativnosti pro celý soubor příznaků.

(ii) Vypočte se míra diskriminativnosti při vynechání jednoho příznaku. Tento krok se provede n-krát (vynechávají se postupně příznaky $v_1, ..., v_n$)

(iii) Z původního souboru příznaků vyloučíme definitivně ten příznak, při jehož vynechání v kroku (ii) došlo k nejmenšímu poklesu míry diskriminativnosti. Takový příznak je zřejmě nejméně informativní.

1.2 Neuronové sítě [NEU]

vyučující:	Doc. Dr. Ing. Vlasta Radová
ročník/semestr studia:	5.ročník/ZS
datum zkoušky:	5. 1. 2017
hodnocení:	1

cíl předmětu (STAG):
Cílem předmětu je seznámit studenty se základními typy umělých neuronových sítí a s možnostmi jejich využití.

1.2.1 Základní umělé modely neuronu, vlastnosti, souvislost s biologickým neuronem.

Lidský mozek se skládá ze 100 miliard neuronů, které mezi sebou komunikují prostřednictvím sítě vazeb. Podnět z receptorů (zrak, sluch, čich, chuť, hmat) je ve formě elektrických impulsů přenášen do centrálního nervového systému, kde je zpracováván.

Biologický neuron

Základní buňka biologických neuronových sítí.

- *soma* (tělo)
- *axon* (výstup): jediný, dlouhý až 60 cm
- *dendrity* (vstupy): krátké do 3 mm, je jich až několik tisíc na neuron
- *synapse* (rozhraní): jednosměrné brány umožňující přenos signálu pouze ve směru axon → dendrita, je jich asi 10000 na neuron

Přenášené signály jsou elektrické impulsy, jejich přenos je ovlivněn uvolňováním budicích (excitátory) a tlumicích (inhibitory) látek v synapsích. Překročí-li hodnota budicích signálů hodnotu tlumicích signálů o určitý *práh*, nastává tzv. *aktivace neuronu* - na jeho výstupu se objeví impuls. Po vygenerování impulsu se neuron na určitou dobu (tzv. období pauzy) dostane do stavu, kdy nereaguje na žádné podněty → chování neuronu lze popsat diskrétně v čase. Období pauzy není pro všechny neurony stejně dlouhé → neurony v mozku pracují asynchronně.

McCullochův-Pittsův model (1943)

Vstupy $x_1, ..., x_n$ mohou nabývat pouze hodnot 0 nebo 1 podle toho, zda je přítomen signál nebo ne. Váhy $w_1, ..., w_n$ nabývají hodnot 1 pro budící signály a -1 pro tlumící signály. Dále je definován práh b. Jedná se tedy o jednoduchý binární model. Váhy jsou pevně nastaveny a práh je rovněž pevně stanoven.

Perceptron (1958)

Diskrétní verze perceptronu byla poprvé prezentována Frankem Rosenblattem. Pro výstup platí:

$$y(k+1) = f\left[\sum_{i=1}^{n} w_i x_i(k) + b\right] = f(w^T \cdot x(k) + b) \tag{1.41}$$

w ... váhový vektor

x ... vstupní vektor

$z = w^T \cdot x(k) + b$ ... aktivační hodnota

$f(\cdot)$ ... aktivační funkce, nejčastěji používané:

- *bipolární binární*: $f(z) = \text{sgn}(z) = \begin{cases} 1 & z \geq 0 \\ -1 & z < 0 \end{cases}$
- *unipolární binární*: $f(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$
- *bipolární spojitá*: $f(z) = \frac{2}{1+e^{-\lambda \cdot z}} - 1$
- *unipolární spojitá*: $f(z) = \frac{1}{1+e^{-\lambda \cdot z}}$
- *lineární*: $f(z) = \lambda \cdot z$

1.2.2 Základní typy neuronových sítí. Způsoby činnosti a učení neuronových sítí.

Umělá neuronová síť vznikne spojením jednotlivých modelů neuronů. Výsledná funkce sítě je určena způsobem propojení jednotlivých neuronů (tzv. topologií sítě), váhami těchto spojení a způsobem činnosti jednotlivých neuronů (aktivačními funkcemi). Mezi základní typy patří:

- *vícevrstvé dopředné neuronové sítě (feedforward nets)*: výstup jedné vrstvy je připojen na vstup následující vrstvy a signál se šíří pouze ze vstupu sítě na její výstup
- *neuronové sítě se zpětnou vazbou (recurrent, feedback nets)*: oproti dopředným sítím se zde signál šíří také z výstupu sítě zpět na její vstup

Fáze činnosti

1. *Fáze nastavování*: cílem je nastavit váhy a prahy sítě tak, aby prováděla požadovanou činnost (předpokládá se, že je dána topologie sítě); Provádí se buď výpočtem (lze jen u naprosto jednoduchých sítí) a nebo učením - trénováním
2. *Fáze pracovní*: síť reaguje na předložené vstupy podle svého předchozího nastavení

Způsoby učení

- *učení s učitelem (supervised learning)*: předpokládáme, že máme k dispozici tzv. trénovací množinu, tj. množinu dvojic [input, desired_output], které jsou síti postupně předkládány. Pro každý předložený vstup neuronová síť vygeneruje skutečný výstup, který se porovná s požadovaným výstupem, a trénovací algoritmus upraví nastavení vah a prahů tak, aby rozdíl mezi skutečným a požadovaným výstupem byl minimální. Každá dvojice se síti předkládá opakovaně - trénování probíhá v trénovacích cyklech (epochách), kdy během jedné epochy jsou sítě předloženy všechny dostupné dvojice. Pořadí výběru dvojic má vliv na výsledek učení. Dobře natrénovaná síť je schopna to, co se naučila, zobecňovat (schopnost generalizace). Trénování může být *dávkové* (váhy a prahy se mění po předložení více dvojic) nebo *sekvenční* (váhy a prahy se mění po každé dvojici).
- *učení bez učitele (unsupervised learning)*: síti jsou předkládány pouze vstupy, informace učitele (požadovaný výstup) chybí. Síť se sama snaží najít zákonitosti ve vstupních datech a nastavit své váhy a prahy tak, aby na podobné vstupy reagovala podobnými výstupy. Podobnost je většinou definována eukleidovskou vzdáleností a dochází tak ke shlukování vstupních dat.

RBF (Radial Basis Function) sítě

Motivaci pro jejich studium lze nalézt v matematice u řešení aproximačních úloh, kdy řešení hledáme ve tvaru lineární kombinace vhodných bazických funkcí. Představena je dvouvrstvá síť bez zpětné vazby. Ve druhé (výstupní) vrstvě jsou neurony s lineární aktivační funkcí. V první (skryté) vrstvě mají všechny váhy hodnotu 1. Každý neuron této vrstvy je reprezentován tzv. *radiální funkcí*, která je funkcí vzdálenosti mezi vstupním vektorem a centroidem příslušného neuronu. Pro výstup j-tého neuronu ve skryté vrstvě tedy platí: $a_j = f(r_j)$, kde $f(\cdot)$ je radiální funkce a r_j je vzdálenost mezi vstupním vektorem x a centroidem j-tého neuronu c_j (obvykle eukleidovská vzdálenost: $r_j = \sqrt{(x - c_j)^T (x - c_j)}$).

Trénuje se pouze druhá (výstupní) vrstva, a to pomocí učení s učitelem. Centroidy první vrstvy lze najít jakýmkoli shlukovacím algoritmem, obvykle se používá *k-means*. Pomocí RBF-sítí lze aproximovat vícerozměrné nelineární funkce a je oproti klasické dopředné síti obecně rychleji natrénovatelná a výsledná aproximace je přesnější.

1.2.3 Algoritmus backpropagation.

Jedná se učení s učitelem, tj. předpokládá se znalost trénovací množiny o P dvojicích $[x_p, u_p]$. Cílem je nastavit váhy a prahy sítě tak, aby výstup sítě byl pro všechny vstupní vektory z trénovací množiny správný, tj. aby byla minimální chyba definována vztahem:

$$E = \frac{1}{2PM} \sum_{p=1}^{P} ||u_p - y_p||^2 = \frac{1}{2PM} \sum_{p=1}^{P} \sum_{m=1}^{M} (u_{pm} - y_{pm})^2 \tag{1.42}$$

y_p ... skutečný výstup sítě pro vstup x_p

u_p ... požadovaný výstup sítě pro vstup x_p

P ... počet trénovacích dvojic

M ... počet výstupů sítě

Váhy a prahy se mění úměrně gradientu chybové funkce, tzn. chyba se z výstupu sítě šíří zpět do jednotlivých vrstev, kde se podle ní mění váhy a prahy.

Značení (předpokládáme síť s jednou skrytou vrstvou):

E ... chyba způsobená nesprávnou činností sítě během jednoho trénovacího cyklu

$f(\cdot)$ ... aktivační funkce ve skryté vrstvě

$g(\cdot)$ ... aktivační funkce ve výstupní vrstvě

Algoritmus:

(i) *Inicializace*

$W^{(1)}(0)$ ... váhová matice pro skrytou vrstvu $(j \times n)$ v kroku 0

$W^{(2)}(0)$ ... váhová matice pro výstupní vrstvu $(m \times j)$ v kroku 0

$B^{(1)}(0)$ ... prahový vektor pro skrytou vrstvu $(j \times 1)$ v kroku 0

$B^{(2)}(0)$ ... prahový vektor pro výstupní vrstvu $(m \times 1)$ v kroku 0

$c > 0$ ... konstanta učení

$E_{max} > 0$ ... maximální povolená chyba, pod kterou se chceme dostat

$ep_{max} > 0$ ... maximální počet trénovacích cyklů (epoch)

(ii) *výpočet výstupu pro vstup x_p v kroku k*

$$y_p = g\left[W^2(k) \cdot f(W^1(k) \cdot x_p + B^1(k))\right] \tag{1.43}$$

(iii) *výpočet chyby predikce pro vstup x_p*

$$E_p = \frac{1}{2}(u_p - y_p)^T(u_p - y_p) \tag{1.44}$$

(iv) *modifikace vah a prahů pro vstup x_p v kroku k*

$$\begin{aligned}
W_{ji}^{(1)}(k+1) &= W_{ji}^{(1)}(k) + c \cdot \sum_{m=1}^{M}(u_{pm} - y_{pm}) \cdot g'(A_{m\cdot}^{(2)}(k)) \cdot W_{mj}^{(2)} \cdot f'(A_{j\cdot}^{(1)}(k)) \cdot x_i \\
B_j^{(1)}(k+1) &= B_j^{(1)}(k) + c \cdot \sum_{m=1}^{M}(u_{pm} - y_{pm}) \cdot g'(A_{m\cdot}^{(2)}(k)) \cdot W_{mj}^{(2)} \cdot f'(A_{j\cdot}^{(1)}(k)) \\
W_{mj}^{(2)}(k+1) &= W_{mj}^{(2)}(k) + c \cdot (u_{pm} - y_{pm}) \cdot g'(A_{m\cdot}^{(2)}(k)) \cdot A_{j\cdot}^{(1)}(k) \\
B_m^{(2)}(k+1) &= D_m^{(2)}(k) + c \cdot (u_{pm} - y_{pm}) \cdot g'(A_{m\cdot}^{(2)}(k))
\end{aligned} \tag{1.45}$$

(v) *zastavovací podmínky* Pokud $E(k) \leq E_{max}$, síť je úspěšně natrénována. Pokud počet trénovacích epoch přesáhl ep_{max}, trénování končí bez ohledu na chybu.

Malá hodnota konstanty učení c zpomaluje proces, ovšem pro velkou hodnotu zase hrozí nebezpečí nestability - je vhodné volit adaptivní c. Strmost aktivační funkce má podobný vliv jako konstanta učení. Algoritmus může skončit v nevyhovujících "mělkých"lokálních minimech chybové funkce (chybová funkce je závislost chyby na parametrech, tj. vahách a prazích sítě). Možná řešení:

- vyzkoušet různé inicializace vah a prahů
- zvýšit počet neuronů ve skrytých vrstvách, popř. počet skrytých vrstev (pouze částečné řešení)
- *momentová metoda*: Jejím smyslem je zabránit tomu, aby se proces trénování zastavil v mělkém lokálním minimu. Pro změnu vah použijeme:

$$w(k+1) = w(k) + \alpha \cdot \Delta w(k-1) - (1-\alpha) \cdot c \cdot \frac{\partial \epsilon}{\partial w} \tag{1.46}$$

  kde $\Delta w(t-1)$ je minulá změna váhy, tj. $\Delta w(t-1) = w(t) - w(t-1)$ a α je momentová konstanta. Analogicky se mění i prahy. Při inicializaci je třeba nastavit $\Delta w(0) = 0$, resp. $\Delta b(0) = 0$. Používá se zejména při dávkovém trénování.

1.2.4 Sítě se zpětnou vazbou. Hopfieldova neuronová síť.

S předpokladem čtvercové matice W se výstup sítě počítá:

$$y_j(k+1) = f\left[\sum_{i=1}^{J} w_{ji} y_j(k) + b_j\right] \tag{1.47}$$

Přenos může být:

- *synchronní*: všechny výstupy přepočítám naráz, tedy pro vstupy v kroku $(k+1)$ používám výstupy z kroku (k)
- *asynchronní*: přepočítávám jeden výstup po druhém, tedy např. pro vstup $y_2(k+1)$ používám výstup $y_1(k+1)$, tj. v každém kroku se vypočítává výstup pouze jednoho neuronu

Proces samovolného přechodu končí buď v *rovnovážných stavech*, kdy $y(k+1) = y(k)$, nebo v *rovnovážných cyklech* tvořených stavy, mezi kterými síť kmitá. Jestliže rekurentní síť pracuje v asynchronním režimu, váhová matice je symetrická a prvky na diagonále jsou nezáporné $\rightarrow$ síť vždy konverguje do rovnovážného stavu. Jestliže rekurentní síť pracuje v synchronním režimu a váhová matice je symetrická, pak síť vždy konverguje do rovnovážného stavu nebo cyklu délky 2.

Hopfieldova síť

Jedná se o jednovrstvou rekurentní síť s neurony s bipolární binární aktivační funkcí, symetrickou váhovou maticí s nulami na diagonále a nulovým prahovým vektorem ($w_{ij} = w_{ji} \forall i, j, i \neq$

$j; \wedge w_{ii} = 0$). Pro výstup i-tého neuronu platí:

$$y_i(k+1) = \operatorname{sgn}\left[\sum_{i=1}^{n} w_{ij} y_j(k)\right] \tag{1.48}$$

$y(k)$ ... výstup sítě v čase (k) = stav sítě v čase (k)

$y(0)$ ... inicializační stav sítě

Po inicializace v čase $k = 0$ přechází síť samovolně z jednoho stavu do druhého (jedná se o dynamický systém). Ke změně stavu sítě může docházet synchronně či asynchronně. Při asynchronním přenosu se neuron, jehož výstup se bude přepočítávat, obvykle vybírá náhodně → *stochastická asynchronní rekurze.*

U Hopfieldovy sítě nedochází k učení, váhy sítě jsou určovány pomocí tzv. *záznamového algoritmu*, během kterého se do sítě zaznamenávají požadované rovnovážně stavy. Pro váhovou matici platí:

$$W = \left[\sum_{p=1}^{P} u_p \cdot u_p^T\right] - P \cdot I \tag{1.49}$$

u_p ... tzv. prototypy, tj. rovnovážné stavy, které mají být do sítě zaznamenány (dimenze n)

P ... počet zaznamenaných prototypů

I ... identická matice řádu n

Pro Hopfieldovu síť lze definovat tzv. *výpočetní energii* ve tvaru:

$$E(y) = -\frac{1}{2} y^T \cdot W \cdot y \tag{1.50}$$

Lze ukázat, že při přechodu sítě z jednoho stavu do jiného se tato energie nezvyšuje a v rovnovážném stavu (resp. cyklu) je minimální. Pro každý rovnovážný stav u existuje tzv. *komplementární stav* u', pro který platí $u' = -u$. Tento stav je rovněž rovnovážným stavem, i když v průběhu záznamového algoritmu nebyl do sítě zaznamenán. Zda proces přechodu sítě z jednoho stavu do jiného skončí v požadovaném nebo komplementárním stavu závisí při asynchronním režimu na pořadí přepočítávání výstupů jednotlivých neuronů → nelze ovlivnit. Do Hopfieldovy sítě lze spolehlivě zaznamenat maximálně $P \leq 0.14 \cdot n$ rovnovážných stavů (n je počet neuronů sítě). Pokud je v síti zaznamenáno rovnovážných stavů více, může proces synchronní i asynchronní rekurze skončit v tzv. *falešném rovnovážném stavu*, který neodpovídá žádnému zaznamenanému ani žádnému komplementárnímu stavu.

V praxi se Hopfieldova síť příliš nevyužívá, právě z důvodů možné existence falešných rovnovážných stavů. Dá se však využít jako rekonstruktor zašuměných dat či pro řešení optimalizačních úloh (např. hledání nejkratší cesty). Princip: Optimalizační úloha se popíše vhodnou funkcí, která se převede do tvaru výpočetní energie sítě. Hopfieldova síť pak samovolně hledá minimum této funkce. Existují i sítě pracující v čase spojitě, ale při jejich analýze je třeba řešit nelineární diferenciální rovnice.

1.2.5 Samoorganizující se sítě.

Samy se snaží objevit zákonitosti a souvislosti ve vstupních datech (tzv. proces samoorganizace), tzn. během učení (bez učitele) se snaží nastavit své váhy a prahy tak, aby na podobné vstupy reagovaly podobnými výstupy → dochází k tzv. shlukování vstupních dat. Míra podobnosti se obvykle posuzuje pomocí Eukleidovské vzdálenosti

$$||a - b|| = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \tag{1.51}$$

Značení:

$S_m(k)$ ... množina vektorů, které jsou zahrnuty ve shluku S_m před přidáním vektoru x

$w_m(k)$ ... centroid shluku S_m před přidáním vektoru x

$P_m(k)$ ... počet vektorů ve shluku S_m před přidáním vektoru x

$S_m(k+1)$ ... množina vektorů, které jsou zahrnuty ve shluku S_m po přidání vektoru x, tzn. $S_m(k+1) = S_m(k) \cup x$

$w_m(k+1)$ ... centroid shluku S_m po přidání vektoru x

$P_m(k+1)$ ... počet vektorů ve shluku S_m po přidání vektoru x, tzn. $P_m(k+1) = P_m(k) + 1$

Předpokládejme, že máme množinu P vektorů dimenze n: $\{x_1, x_2, ..., x_P\}$, které chceme rozdělit do R shluků $S_1, S_2, ..., S_R$, přičemž každý shluk je reprezentován svým centroidem $w_1, w_2, ..., w_R$, pro které platí:

$$w_r = \frac{1}{P_r}\sum_{x_j \in S_r} x_j, \qquad r = 1, ..., R \tag{1.52}$$

kde P_r je počet vektorů ve shluku S_r. Vektor x chceme zařadit do jednoho z R shluků → zařadíme ho do toho shluku S_m, k jehož centroidu je x nejblíže:

$$||x - w_m|| = \min_{r=1,...,R} ||x - w_r|| \tag{1.53}$$

Po přidání vektoru x do shluku S_m se změní poloha centroidu w_m, centroidy ostatních shluků zůstávají beze změny.

Kohonenova síť

Určena pro shlukování vstupních vektorů dimenze n do R shluků. Je to jednovrstvá dopředná síť s R výstupy, n vstupy a nulovým prahovým vektorem. Každý řádek váhové matice je normalizován tak, že má velikost 1, tzn., že pro i-tý řádek váhové matice platí:

$$||w_i|| = \sqrt{\sum_j w_{ij}^2} = 1 \tag{1.54}$$

Síť se učí bez učitele. Učení probíhá na základě tzv. *pravidla vítěze* ("vítěz bere vše") → na základě vstupního vektoru x se změní váhy m-tého neuronu podle vztahu:

$$\hat{w}_m(k+1) = w_m(k) + c \cdot (x^T - w_m(k)) \tag{1.55}$$

kde $c \in \langle 0.1, 0.7 \rangle$ je konstanta učení a m-tý neuron je tzv. *vítěz*, pro kterého platí:

$$w_m \cdot x = \max_{r=1,\ldots,R} w_r \cdot x \tag{1.56}$$

To znamená, že se mění váhy neuronů s nejvyšší aktivační hodnotou, váhy ostatních neuronů se nemění. Po změně vah je třeba váhový vektor m-tého neuronu normalizovat, tzn.

$$w_m = \frac{\hat{w_m}}{||\hat{w_m}||} \tag{1.57}$$

Váhová matice $W(0)$ se inicializuje náhodnými čísly, poté je třeba provést normalizaci každého řádku.

Vlastnosti Kohonenovy sítě:

- Váhy neuronů se při trénování blíží do středu shluků, které reprezentují.
- Síť má pouze jednu vrstvu → lze s ní najít pouze lineárně oddělitelné shluky.
- I v případě lineárně oddělitelných shluků nemusí být tyto shluky vždy nalezeny.
- Váhy měnící se podle výše zmíněného vztahu při konstantní hodnotě c nekonvergují ke konstantním hodnotám. Proto se často využívá proměnlivá konstanta učení $c(t)$, která má po určitou dobu t_p (např. 1000 kroků) konstantní hodnotu c_0, a pak se snižuje, např. dle vztahu:

  $$c(t) = c_0 \cdot e^{-\frac{t-t_p}{t_q}}, \qquad t > t_p \tag{1.58}$$

  kde t_q udává rychlost poklesu. Pokud předem neznáme počet shluků, předpokládáme, že shluků je velké množství. Během trénování pak některé neurony pravděpodobně nebudou měnit své váhy → nemají význam a lze je vypustit.
- Po natrénování sítě bude pro zadaný vstupní vektor největší hodnota na výstupu toho neuronu, jehož váhy reprezentují centroid shluku, ke kterému má daný vstupní vektor nejblíže.

Využití: hledání shluků ve vstupních datech; vektorová kvantizace.

Kohonenova mapa (feature map)

Jedná se o speciální případ Kohonenovy sítě. Neurony jsou uspořádány tak, že tvoří jednorozměrné či dvojrozměrné pole. Učení probíhá bez učitele dle pravidla vítěze. Mění se ovšem nejen váhy vítězného neuronu, ale i váhy neuronů okolních. Pro změnu vah platí:

$$w_i(k+1) = w_i(k) + c \cdot (x^T - w_i(k)), \qquad i \in N_m(k) \tag{1.59}$$

$w_i(k)$ ... i-tý řádek matice W (není normalizovaný)

$N_m(k)$ ... okolí m-tého (vítězného) neuronu v čase k

Pro vítězný neuron s indexem m přitom platí:

$$||x - w_m^T|| = \min_{i=1,\ldots,R} ||x - w_i^T|| \tag{1.60}$$

Řádky váhové matice se nenormalizují! V procesu trénování sítě se velikost okolí zmenšuje. Na začátku trénování se velikost okolí obvykle volí tak, že zahrnuje všechny neurony sítě. Pak se velikost okolí lineárně snižuje až na nulu, kdy okolí obsahuje pouze vítězný neuron. Doba trénování se obvykle volí tak, aby doba, po kterou je velikost okolí rovna 0, byla přibližně 3-krát větší než doba, po kterou docházelo ke snižování velikosti okolí. Inicializace vah se provádí malými náhodnými čísly, obvykle z intervalu $\langle -0.1, 0.1 \rangle$. Konstanta učení c se podobně jako u Kohonenovy sítě volí proměnlivá v čase. Kohonenova mapa se využívá pro redukci počtu příznaků pro klasifikaci či pro vizualizaci vektorů velké dimenze.

1.2.6 Oblasti použití neuronových sítí.

Obecně lze aplikace matematicky formulovat jako aproximaci funkce (tj. optimalizační úlohu):

$$y = f(x, parametry) \tag{1.61}$$

- *Klasifikátory*: úkolem je zařadit vstupní data do skupin (tříd) podle vzájemné podobnosti (nelineární sítě); Vstupem jsou jednotlivé klasifikované obrazy (tj. vektory příznaků), výstupem je informace o zařazení vstupního obrazu do určité třídy. Je-li R počet tříd, potom počet výstupů je sítě je obvykle roven buď R nebo $log_2 R$. Obecně platí, že počet výstupu sítě může být jakékoli číslo z intervalu $\langle log_2 R, R \rangle$. Většinou se ve všech vrstvách využívá některá ze sigmoidálních aktivačních funkcí, která se po natrénování ve výstupní vrstvě nahradí odpovídající binární aktivační funkcí.
- *Aproximátory funkcí (regresory)*: z několika zadaných (naměřených) hodnot je třeba sestavit funkční závislost (nelineární sítě); Jednotlivé vstupy nelineárních sítí představují nezávisle proměnné aproximované funkce, výstupy představují závisle proměnné. Pro aproximaci libovolné funkce postačují 2 až 3 skryté vrstvy neuronů:
  - jedna vrstva rozdělí vstupní prostor nadrovinou (ve 2D přímkou)
  - dvě vrstvy jsou schopny oddělit konvexní oblast
  - tři vrstvy jsou schopny oddělit i nekonvexní oblasti

  Počet skrytých vrstev a počet neuronů v nich má vliv na schopnost sítě zobecňovat (generalizovat) vstupní body trénovací množiny. Velké množství vrstev a neuronů může vést k přetrénování (overfitting) sítě, malé množství k nedotrénování (underfitting) sítě.
- *Paměti a rekonstruktory*: na základě předloženého vstupního obrazu je síť schopna "vybavit si"odpovídající výstupní obraz (asociativní paměť - nelineární sítě), popř. je schopna zrekonstruovat zašuměný vstupní obraz do původní podoby (Hopfieldova síť); Asociativní paměť je paměť adresovaná obsahem, adresou je klíčová hodnota ukládaná s informací. Vstupem sítě je vstupní obraz (tzv. klíč), kterým se přistupuje do paměti, výstupem sítě je příslušný asociovaný obraz. Zapamatovaná informace je v síti uložena v hodnotách vah a

prahů a je rozprostřena po celé síti. Asociativní paměť je schopna vykonávat svou funkci dostatečně správně i při částečném poškození (například při odstranění části neuronů skryté vrstvy). U pamětí adresovaných adresou se obsah z poškozené části ztrácí úplně.

- *Optimalizace*: úkolem je minimalizovat určitou ztrátovou funkci, která je obvykle definována uživatelem (rozvrhování činností, hledání optimální cesty, apod.);
- *Shlukování a redukce příznaků*: objevování shluků ve vstupních datech a redukce počtu příznaků. Základní charakteristikou těchto sítí je tzv. samoorganizace (Hopfieldova síť).

1.3 Zpracování digitalizovaného obrazu [ZDO]

vyučující: Doc. Ing. Miloš Železný Ph.D.
Ing. Petr Neduchal
ročník/semestr studia: 4.ročník/LS
datum zkoušky: 13. 7. 2015
hodnocení: 1
cíl předmětu (STAG):
Porozumět principům zpracování digitalizovaného obrazu a počítačového vidění. Analyzovat vlastnosti obrazové informace a interpretovat tyto informace, navrhnout a vytvořit algoritmus pro zpracování obrazové informace s cílem rozpoznání objektů, jevů či vlastností scény v obraze obsažené.

1.3.1 Bodové jasové transformace.

Jasové korekce

Cílem je stanovit nový jas daného pixelu (bodu), který je funkcí původního jasu a polohy:

$$p_{ij}^{new} = f(i, j, p_{ij}^{old}) \tag{1.62}$$

Nejčastěji se používá matice opravných koeficientů OPR a dostáváme tedy:

$$p_{ij}^{new} = p_{ij}^{old} \cdot OPR_{ij} \tag{1.63}$$

Matici opravných koeficientů získáme *kalibrací* - na snímacím zařízení nesnímáme obraz se známými hodnotami. Z těchto známých správných hodnot (actual) a z naměřených hodnot (measured) vypočteme matici opravných koeficientů:

$$OPR_{ij} = \frac{actual_{ij}}{measured_{ij}} \tag{1.64}$$

Využití se najde například při opravě systematických chyb snímacího řetězce.

Jasové transformace

V tomto případě je funkce stejná pro všechny body (pixely) obrazu, nezáleží na poloze v obrázku:

$$p_{ij}^{new} = f(p_{ij}^{old}) \tag{1.65}$$

Využívá se:

- vyhledávací/převodní tabulky (LookUp Table)
- prahování (segmentace - rozdělení obrazu na oblasti, které souvisejí s reálnými objekty)
- ekvalizace histogramu

1.3.2 Geometrické transformace.

Obecná definice:

$$i' = u(i,j); \qquad j' = v(i,j); \qquad g(i',j') = f(i,j) \tag{1.66}$$

- *vztah známe*: např. rotace, posun, zvětšení, ...
- *vztah hledáme* na základě původního a transformovaného obrazu: např. pomocí korespondence souřadnic na družicovém snímku a na mapě, používají se tzv. vlícovací body

Proces se skládá ze dvou kroků:

1. *plošná transformace*

$$x' = \sum_{r=0}^{m}\sum_{k=0}^{m-r} a_{rk}\cdot x^r \cdot y^k; \qquad y' = \sum_{r=0}^{m}\sum_{k=0}^{m-r} b_{rk}\cdot x^r \cdot y^k \tag{1.67}$$

   Jedná se o polynom m-tého stupně. Např.:

   - bilineární: $x' = a_0 + a_1\cdot x + a_2\cdot y + a_3\cdot xy; \qquad y' = b_0 + b_1\cdot x + b_2\cdot y + b_3\cdot xy$
   - afinní: $x' = a_0 + a_1\cdot x + a_2\cdot y; \qquad y' = b_0 + b_1\cdot x + b_2\cdot y$ - sem patří např. rotace, posun, zvětšení, zkosení, atd.

2. *interpolace jasu*: Nejbližší soused - většinou se postupuje tak, že se udělá inverzní transformace a odečte se nejbližší hodnota v původním obrazu

Geometrické transformace jsou z principu ztrátové. V některých speciálních případech (jako např. otočení o násobek 90°) je informace zachována, obecně to však neplatí. Využití: dálkový průzkum Země, desktop publishing.

1.3.3 Filtrace šumu.

Obecná diskrétní konvoluce:

$$g(i,j) = \sum_{m,n}, \sum_{i,j} f(i-m, j-n) \cdot h(m,n) \tag{1.68}$$

Někdy se setkáme s problémem, jak počítat konvoluci co nejrychleji (např. při požadavku na zpracování v reálném čase). Tento problém částečně řeší tzv. box-algorithm (Šlesinger).

Vyhlazování (filtrace)

Cílem je potlačení šumu v obrazu (např. aditivní šum ν se střední hodnotou μ). Myšlenka je založena na využití n bodů v okolí.

$$f_i' = f_i + v_i; \qquad \frac{f_1 + f_2 + \cdots + f_n}{n} + \frac{\nu_1 + \nu_2 + \cdots + \nu_n}{n} \tag{1.69}$$

Průměrovat můžeme přes více snímků: $g(i,j) = \frac{1}{n}\sum_{k=1}^{n} f_k(i,j)$, kde f_k je obrazová funkce k-tého snímku a n se řádově pohybuje v desítkách $(30-50)$. Alternativně můžeme průměrovat lokálně - v daném okolí, to však přináší nevýhodu rozostření hran (ztratí se detaily) $\rightarrow$ velikost masky by měla být menší, než je nejmenší detail v obraze, který chceme zachovat.

Používané masky:

$$h_1 = \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}; \qquad h_2 = \frac{1}{10}\begin{bmatrix}1&1&1\\1&2&1\\1&1&1\end{bmatrix}; \qquad h_3 = \frac{1}{16}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}; \qquad h_4 = \frac{1}{18}\begin{bmatrix}1&1&1\\1&0&1\\1&1&1\end{bmatrix}$$

V případě h_1 jde o rovnoměrnou masku (Average), h_2 zvýhodňuje středový bod, h_3 zvýhodňuje středový bod a body v hlavních směrech (Gauss) a maska h_4 naopak znevýhodňuje středový bod. Poslední dvě masky mají teoretickou výhodu výpočetní rychlosti, neboť dělení 8, resp. 16, lze v počítači realizovat jako bitový posun a tedy velmi rychle.

Alternativně se dá využít tzv. *maximální zastoupení* (Modus) - výsledkem filtrace je jas, který se v daném okolí vyskytuje nejčastěji. Dále můžeme využít nějakého *výběrového kvantilu* (jako např. Medián). Ten dobře řeší problém výskytu jedné nebo více vychýlených hodnot (např. Salt&Pepper Noise), ovšem je nelineární, porušuje tenké čáry a "trhá"rohy.

1.3.4 Gradientní operátory.

Též se označují jako diferenciální operátory či hranové detektory. Dokáží např. detekovat nespojitosti šedé úrovně v obraze, lze je tedy využít pro segmentaci. Hledáme velikost a směr gradientu.

Gradient ve spojitém případě:

$$|grad\, g| = \sqrt{\left(\frac{\partial g}{\partial x}\right)^2 + \left(\frac{\partial g}{\partial y}\right)^2} \tag{1.70}$$

V diskrétním případě:

$$\Delta x g(i,j) = g(i,j) - g(i,j-1); \qquad \Delta y g(i,j) = g(i,j) - g(i-1,j) \tag{1.71}$$

$$|grad\, g| = \sqrt{(\Delta x g)^2 + (\Delta y g)^2} \tag{1.72}$$

$$\varphi = \operatorname{arctg}\left(\frac{\Delta y g}{\Delta x g}\right) \tag{1.73}$$

Máme tři typy gradientních operátorů:

1. *Aproximace derivací diferencemi*
   - Roberts: $g(i,j) = |f(i,j) - f(i+1,j+1)| + |f(i,j+1) - f(i+1,j)|$
   - Laplace: $h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ → aproximace Laplaceova operátoru ($\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$)

   $$\Delta_x^2 f(i,j) = f(i,j+1) + f(i,j-1) - 2f(i,j) \tag{1.74}$$

   $$\Delta_y^2 f(i,j) = f(i+1,j) + f(i-1,j) - 2f(i,j) \tag{1.75}$$

   $\nabla^2 f(i,j) = \Delta_x^2 f(i,j) + \Delta_y^2 f(i,j) = f(i,j+1) + f(i,j-1) + f(i+1,j) + f(i-1,j) - 4f(i,j)$

   Udává pouze velikost hrany, ale ne její směr. Chceme-li znát i směr hrany, použijeme směrově závislý gradientní operátor.

2. *Srovnání s parametrickým modelem hran* Nejprve označíme směry (např. 8). Vztahy:

   $$|grad\, g| \hat{=} \max_{k=0,\ldots,7}\{g * h_k\} \tag{1.76}$$

   $$\varphi = \operatorname*{argmax}_{k=0,\ldots,7}\{g * h_k\} \tag{1.77}$$

   Používají se masky *Prewitt*, *Sobel*, *Kirsch*.

3. *Průchody nulou 2. derivace obrazové funkce (Marrova teorie)* Hledáme průchody nulou druhé derivace obrazové funkce. Používají se masky pro detekci čáry a masky pro detekci bodu.

1.3.5 Metody segmentace.

Vstupem je intenzitní obraz a výstupem obraz rozčleněný na části, které mají souvislost s objekty reálného světa.

- *Kompletní segmentace*: Vytvořené oblasti jednoznačně korespondují s objekty ve vstupním obraze. Obecně je nezbytná spolupráce s vyšší úrovní zpracování a využívají se znalosti o řešeném problému. V případě, kdy je obraz tvořen kontrastními objekty na pozadí konstantního jasu, přicházejí dobré výsledky i na nižší úrovni zpracování. Např. text, krevní buňky, počítání šroubků.
- *Částečná segmentace*: Vytvořené oblasti jsou homogenní vzhledem k určitým zvoleným vlastnostem (jas, barva, textura, apod.). Oblasti se obecně mohou překrývat a je třeba aplikovat další postupy na vyšší úrovni zpracování. Např. scéna s polem a lesem při pohledu z okna - po segmentaci neodpovídá jedna oblast jednomu objektu.

Další příklady využití: hledání lodí na moři, typické vlastnosti železničních tratí a dálnic (např. maximální zakřivení), řeky se neprotínají

Prahování

Nejjednodušší a nejčastěji používaná metoda segmentace, je nenáročná na hardwarovou realizaci a je nejrychlejší (lze provádět v reálném čase). Důležitá je volba prahu T - úloha, kterou lze obecně jen velmi obtížně provést automaticky. Také lze použít jen pro určitou třídu obrazů (objekty a pozadí musí být jasově snadno rozlišitelné).

$$g(i,j) = \begin{cases} 1 & f(i,j) \geq T \\ 0 & f(i,j) < T \end{cases} \tag{1.78}$$

Možná modifikace:

$$g(i,j) = \begin{cases} 0 & f(i,j) \in D \\ 1 & \text{jinak} \end{cases} \tag{1.79}$$

kde D je množina jasů odpovídajících pozadí (např. u snímků krevních buněk - cytoplazma se jeví v určitém intervalu jasů, pozadí je světlejší, jádro je tmavší).

Dále můžeme použít prahovaní více prahy či poloprahování (odstraníme pozadí, v objektech však zachováme rozložení jasů). Používá se při vizuálním hodnocení výsledků člověkem. $f(i,j)$ nemusí být pouze jasová funkce (ale také třeba hodnota gradientu, hloubková mapa, barva, ...).

Metody určování prahu:

- *Histogram*: Hledáme lokální minimum mezi dvěma největšími dostatečně vzdálenými lokálními maximy.
- *Procentní prahování*: Máme apriorní znalost o tom, kolik procent plochy obrazu pokrývají objekty (např. průměrné pokrytí plochy stránky textem se pohybuje okolo 5%). Práh potom nastavíme tak, aby právě tolik procent obrazových bodů mělo barvu objektů, zbytek barvu pozadí.

Segmentace na základě detekce hran

Hrany jsou místa obrazu, kde dochází k určité nespojitosti, většinou v jasu, ale také v barvě, textuře, apod. Obraz hran vznikne aplikací některého hranového operátoru (Canny). Obvykle jen málo míst v obraze má nulovou hodnotu velikosti hrany (šum), je třeba tedy např. prahováním potlačit nevýrazné hrany a zachovat pouze ty významné. Často se využívá apriorní informace o možné poloze hran, popř. znalosti koncových bodů.

Segmentace narůstáním oblastí (region growing)

Metoda uplatnitelná v obrazech se šumem, kde se obtížně hledají hranice. Významnou vlastností je homogenita. Kritérium homogenity je založeno na jasových vlastnostech, komplexnějších způsobech popisu nebo dokonce na vytvářeném modelu segmentovaného obrazu. Většinou pro oblasti požadujeme splnění těchto podmínek:

- $H(R_i) = \text{TRUE}$, $i = 1, 2, ..., l$
- $H(R_i \cup R_j) = \text{FALSE}$, $i, j = 1, 2, ..., l$; $i \neq j$; (R_i soused R_j)

l ... počet oblastí

R_i ... jednotlivé oblasti

$H(R_i)$ ... dvouhodnotové vyjádření kritéria homogenity, oblasti musí být homogenní a maximální

Spojování oblastí: Na začátku každý obrazový element (2x2, 4x4, 8x8, ...) představuje samostatnou oblast, dále spojujeme vždy dvě sousední oblasti, pokud vyhovuje kritérium homogenity. Výsledek závisí na pořadí spojování. Proces končí v okamžiku, kdy nelze spojit žádné dvě oblasti.

Štěpení a spojování (split and merge): Využívá pyramidální reprezentaci obrazu. Oblasti jsou čtvercové a odpovídají elementu dané úrovně pyramidální datové struktury. Na počátku určíme nějaké počáteční rozložení obrazu. Platí-li pro obraz R i-té úrovně pyramidální struktury $H(R) = \text{FALSE}$ (oblast není homogenní), rozdělíme R na 4 oblasti $(i + 1)$. úrovně. Existují-li sousední oblasti R_i a R_j takové, že $H(R_i \cup R_j) = \text{TRUE}$, spojíme R_i a R_j do jedné oblasti. Nelze-li žádnou oblast spojit ani rozdělit, algoritmus končí.

Segmentace srovnáváním se vzorem (template matching)

Úloha má za úkol nalézt známé objekty (vzory) v obraze. Kromě hledání objektu této metody lze využít také např. pro srovnávání dvou snímků z různých míst či k určování relativního pohybu objektů. Nepřítelem je zde opět šum. Jako míru souhlasu většinou využíváme vzájemnou

korelaci:

$$C_1(u,v) = \frac{1}{\max\limits_{i,j\in V}|f(i+u,j+v) - h(i,j)|}$$
$$C_2(u,v) = \frac{1}{\sum\limits_{i,j\in V}|f(i+u,j+v) - h(i,j)|} \tag{1.80}$$
$$C_3(u,v) = \frac{1}{\sum\limits_{i,j\in V}|f(i+u,j+v) - h(i,j)|^2}$$

Testujeme souhlas obrazu f se vzorem h umístěným v poloze (u, v). Pro každou polohu vzoru h v obraze f určíme hodnotu míry souhlasu vzoru s danou částí obrazu. Problémy nastanou, pokud se vzor v obraze vyskytuje natočený, s jinou velikostí nebo s geometrickým zkreslením. Můžeme to řešit zkoušením více případů či hledáním nejprve menších částí vzoru. Metodu lze urychlit zrychleným prováděním testů v hrubším rozlišení a v místě lokálního maxima pak přesným doměřením.

1.3.6 Matematická morfologie.

Oblast analýzy obrazu, která se opírá o teorii bodových množin (binární obraz, obraz s více úrovněmi jasu). Ve středu pozornosti je tvar objektů → identifikace tvaru, optimální rekonstrukce tvaru, který je porušen. Obrazy jsou nejprve předzpracovány např. metodami segmentace jsou nalezeny objekty (binární obraz) a poté jsou použity morfologické postupy. Použití především pro: odstranění šumu, zjednodušení tvaru objektů, zdůraznění struktury objektů (kostra, ztenčování, zesilování, výpočet konvexního obalu, označování objektů), popis objektů číselnými charakteristikami (plocha, obvod, projekce)

Relace s menší bodovou množinou (strukturním elementem): Morfologickou transformaci si představíme, jako bychom pohybovali strukturním elementem systematicky po celém obraze. Bod obrazu, který se shoduje s počátkem souřadnic strukturního elementu, nazýváme okamžitý bod, a právě do něj zapíšeme výsledek relace. Ke každé morfologické transformaci $\Phi(x)$ existuje duální transformace $\Phi^*(x)$. Mezi základní transformace patří posunutí, dilatace, eroze, otevření a uzavření

Dilatace a eroze

Dilatace $\oplus$ skládá body dvou množin pomocí vektorového součtu

$$X \oplus B = \{d \in E^2; d = x + b;\, x \in X,\, b \in B\} \tag{1.81}$$

Nejčastěji používán strukturní element (3x3). Objekty se rozrostou o jednu slupku na úkor pozadí, díry a zálivy tloušťky 2 se zaplní.

Eroze $\ominus$ skládá dvě bodové množiny s využitím rozdílu vektorů, je duální (ale ne inverzní) transformací k dilataci.

$$X \ominus B = \{d \in E^2; d + b \in X;\, \forall b \in B\} \tag{1.82}$$

Nejčastěji se opět používá element (3x3). Zmizí objekty (čáry) tloušťky 2 a osamělé body, objekty se zmenší o 1 slupku. Odečteme-li od původního obrazu jeho erozi, dostaneme obrysy objektu.

Otevření a uzavření

V obou případech jde o kombinaci dilatace a eroze a výsledný obraz obsahuje méně detailů.

Otevření $\circ$ je eroze následovaná dilatací:

$$X \circ B = (X \ominus B) \oplus B \tag{1.83}$$

Oddělí objekty spojené úzkou šíjí, odstraní malé detaily.

Uzavření $\bullet$ je naopak dilatace následovaná erozí:

$$X \bullet B = (X \oplus B) \ominus B \tag{1.84}$$

Spojí objekty, které jsou blízko u sebe, zaplní malé díry a úzké zálivy.

Pokud se obraz po otevření/uzavření elementem B nezmění, říkáme, že je otevřený/uzavřený vzhledem k B. Obě transformace jsou *idempotentní*, tj. opakovaným použitím těchto operací se nezmění výsledek.

Skelet

Skelet $S(Y)$ je množina bodů - středů kružnic, které jsou obsaženy v Y a dotýkají se Y alespoň ve dvou bodech. Lze vytvořit pomocí erozí a dilatací, ale potom může skelet být tlustší než jeden bod. Často se skelet nahrazuje množinou zpracovanou sekvenčním homotopickým zpracováním (hit or miss transformace).

Aplikací funkcí *ztenčování*, resp. *zesilování*, dojde k okleštění skeletu o izolované body a krátké čáry z konců skeletu. V konečném stavu skelet obsahuje pouze uzavřené křivky.

Kapitola 2

Teorie řízení [TŘSZ]

2.1 Lineární systémy 1-2 [LS1], [LS2]

vyučující:	Doc. Ing. Jiří Melichar, CSc. Ing. Martin Čech, Ph.D. Ing. Jiří Mertl, Ph.D.
ročník/semestr studia:	2.ročník/ZS-LS
datum zkoušky:	X. 1. 2013/X. X. 2013
hodnocení:	1/2

cíl předmětu (STAG):

LS1: Student by měl získat přehled o typech, struktuře a chování reálných dynamických systémů, obeznámit se s metodikou tvorby matematických modelů reálných dynamických systémů a s metodami analýzy jejich vlastností a chování v časové i frekvenční oblasti. Student by měl také porozumět základním principům řízení dynamických systémů a metodám pro získávání potřebných dat z reálných procesů.

Cílem předmětu LS2 je, aby student:

- získal přehled o klasických regulačních úlohách, o struktuře regulačních obvodů a o základních typech dynamických i nedynamických regulátorů;
- dokázal analyzovat reálnou regulační úlohu v její celistvosti, uměl formulovat požadavky na kvalitu regulace v časové i frekvenční oblasti při současném respektování všech omezení;
- byl schopen použít vhodné metody pro návrh spojitých i číslicových regulátorů a získávat potřebná data z reálného procesu;
- byl schopen analýzy nelineárních dynamických systémů a základní orientace v problémech jejich řízení.

2.1.1 Matematické modely spojitých a diskrétních lineárních dynamických systémů.

2.1.2 Linearizace nelineárních dynamických systémů, rovnovážné stavy. Harmonická linearizace.

2.1.3 Vlastnosti lineárních dynamických systémů. Řiditelnost, pozorovatelnost, kriteria. Vnitřní a vnější stabilita, kriteria.

2.1.4 Časové a frekvenční odezvy elementárních členů regulačních obvodů.

2.1.5 Základní typy spojitých a diskrétních regulátorů (P,PI,PID, stavové regulátory a stavové regulátory s integračním charakterem), popis, vlastnosti.

2.1.6 Struktura regulačních obvodů s jedním a dvěma stupni volnosti, přenosy v regulačním obvodu, princip vnitřního modelu.

2.1.7 Problém umístitelnosti pólů a nul nedynamickými a dynamickými regulátory. Požadavky na umístění pólů, konečný počet kroků regulace.

2.1.8 Požadavky na funkci a kvalitu regulace (přesnost regulace, dynamický činitel regulace, kmitavost, robustnost ve stabilitě a j.), omezení na dosažitelnou kvalitu regulace.

2.1.9 Metoda geometrického místa kořenů, pravidla pro konstrukci a využití při syntéze regulátorů, příklady.

2.1.10 Přístup k syntéze regulátorů v klasické teorii regulace, klasické metody, heuristické metody.

2.1.11 Deterministická rekonstrukce stavu, stavový regulátor s rekonstruktorem stavu.

2.1.12 Ljapunovova teorie stability. Ljapunovova rovnice.

2.2 Teorie odhadu [TOD]

vyučující:	Prof. Ing. Miroslav Šimandl, CSc.
	Ing. Jindřich Duník, Ph.D.
ročník/semestr studia:	3.ročník/ZS
datum zkoušky:	28. 4. 2014
hodnocení:	1

cíl předmětu (STAG):
Cílem předmětu je obeznámit studenty s možnostmi odhadu parametrů, náhodných veličin a náhodných procesů v podmínkách neurčitosti z apriorních informací a měřených dat.

2.2.1 Problémy odhadu, základní etapy vývoje teorie odhadu, náhodné veličiny, náhodné procesy a jejich popis, stochastický systém.

2.2.2 Optimální odhad ve smyslu střední kvadratické chyby. Odhad ve smyslu maximální věrohodnosti.

2.2.3 Jednorázové a rekurzivní odhady.

2.2.4 Odhad stavu lineárního diskrétního systému – filtrace (Kalmanův filtr).

2.2.5 Úlohy odhadu stavu lineárního diskrétního stochastického systému – predikce a vyhlazování.

2.2.6 Odhad stavu lineárního systému se spojitým či diskrétním měřením (Kalman-Bucyho filtr).

2.3 Optimální systémy [OPS]

vyučující:	Ing. Miroslav Flídr, Ph.D.
	Ing. Ivo Punčochář, Ph.D.
ročník/semestr studia:	4.ročník/LS
datum zkoušky:	15. 7. 2015
hodnocení:	3

cíl předmětu (STAG):

Cílem předmětu je seznámení studentů s různými typy optimalizačních úloh. Studenti se naučí řešit jednak základní statické optimalizační úlohy tak především úlohy optimalizace dynamických systémů. Důraz je kladen především na pochopení řešení následujících problémů:

- časově optimální řízení;
- Pontrjaginův princip minima;
- dynamické programování a Bellmanova optimalizační rekurze;
- lineárně - kvadratická úloha optimálního řízení.

2.3.1 Optimální programové řízení diskrétních dynamických systémů. Formulace úlohy. Hamiltonova funkce. Nutné podmínky pro optimální řízení.

2.3.2 Optimální programové řízení spojitých dynamických systémů. Formulace úlohy. Hamiltonova funkce. Nutné podmínky pro optimální řízení. Podmínky transverzality. Pontrjaginův princip minima.

2.3.3 Deterministický diskrétní systém automatického řízení. Princip optimality. Bellmanova funkce. Bellmanova optimalizační rekurze.

2.3.4 Syntéza optimálního deterministického systému automatického řízení pro diskrétní lineární řízený systém a kvadratické kritérium. Formulace a řešení. Asymptotické řešení a jeho stabilita.

2.3.5 Deterministický spojitý systém automatického řízení. Kontinualizace Bellmanovy optimalizační rekurze.

2.3.6 Optimální stochastický systém automatického řízení. Strategie řízení. Bellmanova funkce a Bellmanova optimalizační rekurze.

2.3.7 Syntéza optimálního systému automatického řízení pro lineární gaussovský řízený systém a kvadratické kritérium. Formulace a řešení. Separační teorém.

2.4 Adaptivní systémy [AS]

vyučující:	Ing. Jindřich Duník, Ph.D.
	Ing. Ladislav Král, Ph.D.
ročník/semestr studia:	5.ročník/ZS
datum zkoušky:	12. 12. 2016
hodnocení:	1

cíl předmětu (STAG):
Cílem předmětu je obeznámit studenty s adaptivními systémy automatického řízení a adaptivními systémy zpracování signálů.

2.4.1 Základní přístupy k syntéze adaptivních řídicích systémů, schematické vyjádření, srovnání s předpoklady a návrhem standardních regulátorů.

2.4.2 Adaptivní řízení s referenčním modelem, MIT pravidlo, využití Ljapunovovy teorie stability.

2.4.3 Samonastavující se regulátory, charakteristika a základní přístupy k návrhu bloku řízení, přiřazení pólů, diofantické rovnice, minimální variance.

2.4.4 Samonastavující se regulátory, charakteristika a základní přístupy k návrhu bloku poznávání, parametrické metody odhadu.

2.4.5 Adaptivní systémy na zpracování signálu, adaptivní prediktor, adaptivní filtr, analogie se samonastavujícími se regulátory.

2.4.6 Adaptivní řízení a strukturální vlastnost stochastického optimálního řízení, duální řízení, neutralita, separabilita, ekvivalence určitosti.

Kapitola 3

Aplikovaná kybernetika [AKSZ]

3.1 Umělá inteligence [UI]

vyučující:	Prof. Ing. Josef Psutka, CSc.
	Ing. Aleš Pražák, Ph.D.
ročník/semestr studia:	2.ročník/ZS
datum zkoušky:	X. X. 2012
hodnocení:	1

cíl předmětu (STAG):
Cílem předmětu je seznámit studenty se základními problémovými oblastmi umělé inteligence (UI) a naučit je aplikovat vybrané metody řešení úloh, reprezentace znalostí v UI a hraní her.

3.1.1 Metody řešení úloh v UI

3.1.2 Logické formalizmy pro reprezentaci znalostí. Predikátový počet 1. řádu. Rezoluční metoda.

3.1.3 Produkční systém. Báze znalostí a báze dat. Dopředné a zpětné šíření.

3.1.4 Síťové formalizmy pro reprezentaci znalostí. Sémantické sítě. Rámce. Scénáře.

3.1.5 Metody hraní her v UI. Procedura minimax, alfa-beta prořezávání.

3.2 Modelování a simulace 1 [MS1]

vyučující:	Ing. Václav Hajšman, Ph.D.
	Ing. Jindřich Liška, Ph.D.
	Ing. Miloš Fetter
ročník/semestr studia:	2.ročník/ZS
datum zkoušky:	X. X. 2012
hodnocení:	1

cíl předmětu (STAG):
Cílem předmětu je seznámit studenty se základními principy modelování dynamických systémů.

3.2.1 Systém, model, modelování, simulace, systémová analýza.

3.2.2 Modelování systému diskrétních událostí, diskrétní simulace.

3.2.3 Simulační experiment, studie, analýza rizika, náhoda v simulačních úlohách.

3.2.4 Modelování v netechnických oborech (kompartmenty, buněčné automaty, ...).

3.2.5 Konstrukce modelů na základě měření, zpracování signálu v časové, frekvenční a časo-frekvenční oblasti, modely periodických procesů.

3.2.6 Modely vibrací a kmitání, experimentální modální analýza.

3.2.7 Generování náhodných čísel, metoda Monte Carlo a odhad přesnosti simulačních výsledků.

3.3 Programové prostředky řízení [PP]

vyučující:	Ing. Pavel Balda, Ph.D.
ročník/semestr studia:	3.ročník/LS
datum zkoušky:	X. X. 2014
hodnocení:	1

cíl předmětu (STAG):

Cílem předmětu je naučit studenty aplikovat některé vybrané techniky programování řídicích a informačních systémů především prostředky jazyka C#. V rámci předmětu je podána klasifikace operačních systémů a jejich základní vlastnosti. Dále je vysvětlena hierarchie programového vybavení typických řídicích systémů od čidel a akčních členů až po podnikové systémy.

3.3.1 Architektura podnikových řídicích systémů; používané programovací jazyky.

3.3.2 Architektura .NET Frameworku; řízený modul, metadata, běh řízeného kódu.

3.3.3 Jazyk C Sharp: hodnotové a referenční typy; jednoduché typy, implicitní konverze; výrazy a operátory; příkazy; výjimky.

3.3.4 Jazyk C Sharp: Členy a přístup k nim; jmenné prostory; třídy, metody, vlastnosti, konstruktory, destruktory; struktury; pole; delegáty; atributy.

3.3.5 Softwarové komponenty: DLL, RPC, COM; interface; OPC.

3.3.6 Operační systémy: procesy a thready, synchronizace, deadlock, inverze priorit; správa paměti; vstupně-výstupní systém, programované vstupy/výstupy, přerušení, DMA, ovladače zařízení; souborové systémy.

3.3.7 Operační systémy reálného času: statické a dynamické plánovací algoritmy.

3.3.8 Struktury vzdálených a virtuálních laboratoří.

3.4 Převodníky fyzikálních veličin [PFV]

vyučující:	Ing. Liber Jelínek Ph.D.
ročník/semestr studia:	4.ročník/LS
datum zkoušky:	16. 6. 2016
hodnocení:	2

cíl předmětu (STAG):

Cílem předmětu je seznámit studenty se základními principy, vlastnostmi a modely senzorů a akčních členů pro potřeby automatizace, monitorování a diagnostiky.

3.4.1 Struktura a parametry senzorů pro automatizaci, statické a dynamické modely a chyby, metody snižování chyb senzorů.

3.4.2 A/D a D/A převodníky, obvody pro úpravu signálů, frekvenční filtry.

3.4.3 Senzory teploty a tepla, obvody pro měření odporu, kapacity, indukčnosti a frekvence.

3.4.4 Senzory polohy a vzdálenosti (odporové, indukční, kapacitní, ultrazvukové, optické).

3.4.5 Senzory síly, hmotnosti, deformace, tlaku, rychlosti, zrychlení a vibrací (tenzometrické, piezoelektrické, kapacitní a elektrodynamické).

3.4.6 Senzory průtoku, množství, hustoty, viskozity, koncentrace a chemického složení.

3.4.7 Elektrické akční členy a jejich budiče (stejnosměrné, střídavé, krokové motory, PWM zesilovače, frekvenční měniče).

3.4.8 Hydraulické a pneumatické akční členy (pracovní a řídicí mechanizmy a zdroje tlakového média).