Diagnostika stavu součástí helikoptéry

Petr Sládek <sladek@asix.cz>, CTU FEE id: sladep1

1. Úvod do problematiky

1.1 stručný popis:

- 8 senzorů vibrací připevněných na zkoumaných součástech
- dostupné vzorky dat pro pro simulované poruchy a bezprouchový stav při různých kroutících momentech

1.2 předběžná manuální kontrola dat

- v časové doméně se jeví signál jako ustálený, nevyskytují se v něm časově lokalizované kmity (např. jako v řeči), které by poukazovaly na převední do wavletové domény
- z předchozícho bodu bude tedy použita fourierova transformace (reálně FFT) pro stanovení frekvenčních složek
- spektra FFT mají následující povahu:
    a) vždy existuje dominantní peak, jehož pozice není ovlivněna přenášeným momentem, tj. jedná se o otáčkovou frekvenci
    a) u některých sond je počet takových peaků necitlivých vůči momentu větší (v blízkosti senzorů zřejmě výskyt ozubených kol)
    b) ostatní peaky vykazují velmi velikou závislost na momentu, jak polohou tak amplitudou. Z povahy signálu odhaduji, že tyto komponenty budou mít přibližně stejný "akustický" výkon, ale velmi se mění jeho spektrum.

Z bodů a-c) vyplývá nutnost segmentace spekter do tzv. výkonových segmentů (frekvečnách pásem). Každému segmentu bude po výpočtu přiřazen logaritmus součtu kvadrátu amplitud (to odpovdá fyzikálně logaritmu výkonu*2).

1.3 "nevyslovený" předpoklad řešení

Jakákoliv změna (vada) vede ke změně emitovaného akustického výkonu soustrojím ve vybraných frekvenčních pásmech nebo celkového výkonu.

1.4 návrh segmentace spekter

Jak už bylo uvedno výše, každému segmentu s je přiřazen příznak:

$$f_s = \log \sum_{i=Ns}^{Ns+N-1} FFT_i^2(signal)$$

Důvod proč je použit logaritmus - umožňuje zachování "detailů" i nízkovýkonových signálů. Dále byla stejně provedena normalizace příznakového prostoru pro účely selekce příznaků, takže smysl logaritmu se vytrácí. W-extrakce (fisherm()) také

normalizuje (tvoří afinní zobrazení).

2. Analýza dat

2.1 Statistická analýza příznakového prostoru:

Tvrzení 2.1. Příznaky jsou ovlivněny dvěma typy šumu:
- náhodný, jehož zdroj je neznámý (předpoklad gaussian white noise),
- závislý na přenášeném momentu, fuknční závislost je neznámá a není znám ani moment (neznámé rozdělení)

2.2 Separabilita příznakového prostoru - selekce příznaků:

Dále popsaná metoda má za úkol odhadnout separabilitu množin (shluků) příznakového prostoru.

Tvrzení 2.2. Metoda založená na odhadu středních hodnot a rozptylu deformuje shluky (množiny) na hyperelipsoidy, v horším případě na hyperkoule, což má za následek vysokou deformaci množin, následně pak pesimistický odhad na separabilitu shluků příznaků.

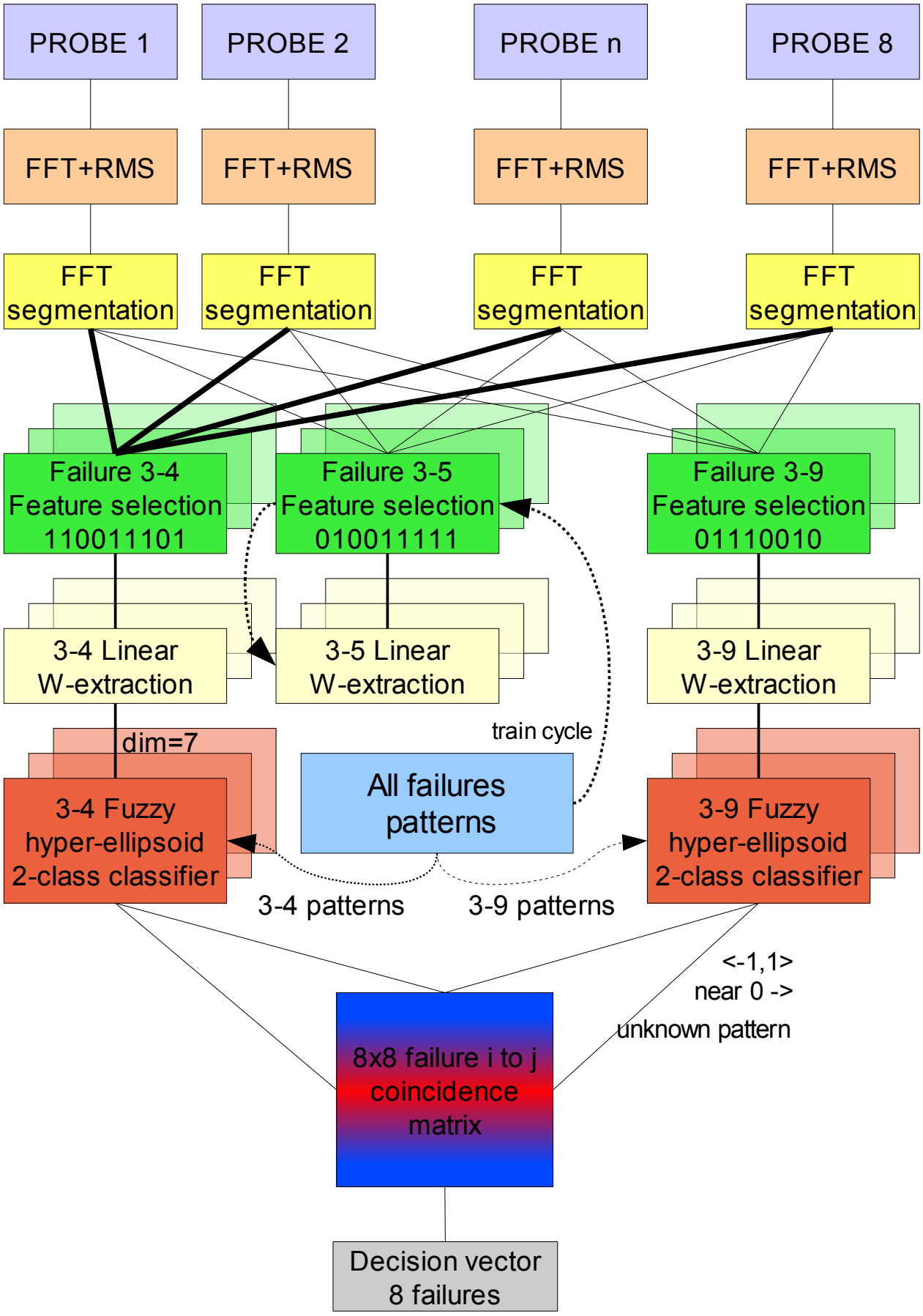
Tvrzení 2.3. Metoda odhadu separability má za úkol zodpovědět "jak moc jsou množiny v konjunkci" - tj. kolik prvků mají společných a kolik ne. Takovýto pohled na věc určuje i pravěpodobnost špatného zařazení klasifikátorem. Přímo z lokálních odhadů hustot lze řídit výslednou chybovost.

V principu jde o definici měřítka, kolik "hyperobjemu" je společného (v konjunkci) a kolik ne. Analyticky modelovat množiny a vypočítávat jejich půniky je příliš složitá záležitost v N dimenzionálním prostoru, proto jsem navrhnul řešení "hvězdokup".

2. Konstrukce diganostického systému

Tato část trochu předbíhá, ale názorně ukáže celou koncepci. Důvod proč byla vybrána vyplyne z následujících odstavců.

- obrázek na další straně -


PROBE 1
PROBE 2
PROBE n
PROBE 8
FFT+RMS
FFT+RMS
FFT+RMS
FFT+RMS
FFT segmentation
FFT segmentation
FFT segmentation
FFT segmentation
Failure 3-4 Feature selection 110011101
Failure 3-5 Feature selection 010011111
Failure 3-9 Feature selection 01110010
3-4 Linear W-extraction
3-5 Linear W-extraction
3-9 Linear W-extraction
dim=7
train cycle
All failures patterns
3-4 Fuzzy hyper-ellipsoid 2-class classifier
3-9 Fuzzy hyper-ellipsoid 2-class classifier
3-4 patterns
3-9 patterns
<-1,1>
near 0 ->
unknown pattern
8x8 failure i to j coincidence matrix
Decision vector 8 failures

3. Selekce příznaků metodou "hvězdokup"

Myšlenka metody je prostá - odhadnout kdy "hvězda" (bod) patří do hvězdokupy (shluku) a kdy ne. Aplikací na všechny body v N-dimenzionálním prostoru se pak odhadne celková konjunkce množin. Základem je výpočet lokálních "mezihvězdných" vdáleností z např. NN-nejbližších sousedů a tím odhadnout lokální "rozptyl". NN by mělo být v desítkách (10-50) pro relevantní statistický výsledek.

Poznámka. Symbolem --> je označna dílčí metoda, která byla implementována.

3.1) normalizace pro každé dvě zkoumané množiny

Odstraní se offset a normalizuje se příznakový prostor tak, aby každý příznak byl v rozsahu <0,1>

- **`file: featnorm.m`**

3.2) stanoví se vnitrotřídní odlehlost bodů pro třídy C^k, $k=(1..N)$:

- výběr a výpočet metrik pro body c_i, c_j, které náleží do $\boldsymbol{C}^k$, $\boldsymbol{M}^k = metrics(c^k_i, c^k_j)$, $i!=j$
- **`file: metrics.m`**

  {
  - minkovského,
  --> eukleidovská $dist(\boldsymbol{C}^k)$,
  - city block $linkdist(\boldsymbol{C}^k)$,
  - hexagonální síť,
  - etc.

  }

- výpočet vnitrotřídní odlehlosti $\boldsymbol{ICD}^k = icd(\boldsymbol{M}^k)$, vrací jeden prvek nebo vektor.
- **`file: icd.m`**

  {
  - aritmetický průměr všech spočtených metrik pro body c^k_i, c^k_j
  - mocninný průměr všech spočtených metrik pro body c^k_i, c^k_j
  --> aritmetický průměr NN-nejbližších sousedů pro každý bod zvlášť
  - určení NN-nejbližších sousedů, výpočet střední hodnoty, a následně rozptylu - tzv. lokální mu, sigma.
  - etc.

  }

Poznámka. NN určuje kvalitu odhadu rozptylu vnitrotřídní odlehlosti a zároveň potlačuje efekt zvýšení vnitrotřídní odlehlosti díky veliké, i když hustě obsazené množiny.

3.2) výpočet odhadu konjunkce množin C^k

- výpočet konjunkce *inset()* $\boldsymbol{C}^l$ a $\boldsymbol{C}^m$
  {
  --> **"ostré" množiny (nefuzzy):**
  Jestliže je vzdálenost bodů *metrics*(c^l_i, c^m_j) menší než $q(icd^{\,l}_i + icd^{\,m}_j)$ body c^l_i, c^m_j jsou v kojunkci, náleží do obou množin *inset*(c^l_i, c^m_j) vrací 1, opačně 0. Kde q určuje s jakou pravděpodobností se protnou - nutno určit z odhadu rozptylu.

  --> **fuzzy množiny:**
  `file fuzzyinset.m`
  Možnost adjustovat "rozplyzlost" množiny pomocí q. Default q=1.
  $\boldsymbol{COM}$=*fuzzyinset()=ceiling_b0t1*($q(((icd^{\,m}_j + icd^{\,l}_i)$-*metrics*($c^l_i$, c^m_j))/($icd^{\,m}_j + icd^{\,l}_i$))
  matice $\boldsymbol{COM}$ je l x m obsahuje hodnoty <0,1> které specifikují jak moc jsou body společné oběma množinám.
  }

- kvantifikace celkové konjunkce
  {
  -->
  `file conjthresh.m`
  Celková konjunkce dvou množin $\boldsymbol{C}^l$ a $\boldsymbol{C}^m$ se určí jako
  sum(sum($\boldsymbol{COM}$))/min{size($\boldsymbol{COM}$)}
  počet řádků udává počet známých bodů množiny $\boldsymbol{C}^l$ a počet bodů z $\boldsymbol{C}^m$.
  Toto škálování je nutné z důvodu nestejného počtu bodů v $\boldsymbol{C}^l$ a $\boldsymbol{C}^m$.
  Takováto definice nezachovává "objemový" průnik. Je jen odhadem průniku, ale důležité pro další je:
  Nikde v selekci nepočítáme s nějakou fixní hodnotou konjunkce, tažepro naše účely postačí tato definice. Viz dále - selekce.

  - pro lepší odhad kojnukce by bylo vhodné zavést ještě čtvercové matice COM_$\boldsymbol{C}^l$, COM_$\boldsymbol{C}^m$. Ty by svými prvky udávaly jak moc jsou v konjunkci vlastní prvky množin $\boldsymbol{C}^l$ a $\boldsymbol{C}^m$ a s pomocí COM by se se pak daly odhadnout společné body a tím normalizovat metriku konjukce tak, aby lépe odpovídala "objemovému" průniku. Zatím nerozpracováno.
  }

3.3) selekce příznaků

- **`file: featselect.m`**
- --> Postupně se bude "vypínat" příznak po příznaku - tzv. putující 0 (11..1011..1). Jestliže se celková konjunkce zvýší nebo zůstane stejná, příznak nelze zahodit. Jestliže se konjunkce sníží oproti použití všech příznaků, ukazuje to na fakt, že přínak nenese informaci (šum). Tato selekce nerozezná redundantní příznak, ale nezahodí ho. Následně může proběhnout extrakce těchto redundatních příznaků, zda nejdou sloučit *W*-extrakcí.
  Jelikož hodnotíme pouze změny konjunkce a nemění se v dané selekci počet bodů množin, lze použít definici celkové konjunkce z bodu 3.2.

- Druhý typ algoritmu (nepoužitý) je založen na putující řadě nul (000..0111..1), taková selekce ale musí rozlišovat mezi snížením a stejnou konjunkcí, snížení = příznak nese separující informaci, stejná konjunkce = redundantní příznak nesoucí informaci, musí být použit pro lineární *W*-extrakci pro zachování robustnosti systému. V každém cyklu se jako referenční konjunkce bere z příznaků, které byly selektovány putující řadou nul. (tj. v každém cyklu se konjunkce mění oproti výše uvedenému algoritmu, kde je stejná "overall").

Poznámka. Selekce příznaků byla realizována vždy mezi dvěma množinami, kdy každá reprezentovala jednu poruchu. Je to z důvodů snížení výpočetní složitosti.

5) Lineární *W*-extrakce příznaků

Bohužel jsem nezaznamenal na přednáškách ani v knize [1] smysluplnost použití lineární reálné *W*-extrakce. Smysluplná je právě tehdy, když nezhorší separabilitu množin:

Tvrzení 5.1. 1 dimenzonální* ***W***-extrakci příznaků ***f*** (sloučení příznaků vektoru ***f*** do jednoho příznaku) multimnožin $\boldsymbol{M}^1..\boldsymbol{M}^N$ s reálnými koeficenty lze provést právě tehdy, když existují 100% separující rovnoběžné nadroviny k množinám $\boldsymbol{M}^1..\boldsymbol{M}^N$. Kolmá nadrovina k separujícím je transformační ***W***.

*platí pro ***W*** je matice 1 x length(***f***), dále lze rozšířit na více dimenzí tj. ***W*** je matice *n* x length(***f***).

Jelikož mě dostatečně rychle nenapadlo jak udělat jednoduchý algoritmus, který řeší problém tvrzení 5.1 (adept je modifikace SVM na více rovnoběžných nadploch), použil jsem nepříliš dobré fisherovo mapování založené na vnitrotřídní a mezitřídní odlehlosti (Intra/inter class distance extraction viz [1] p.209). Redukce byla provedena na fixní hodntu 7 dimenzí.

V případě, že bychom použili extrakci založenou na tvrzezní 5.1, lze ji aplikovat postpně na všechny příznaky (n), pro které je dílčí extrakce možná. Tím získáváme m-dimenzionální redukovaný prostor ($m<=n$).

Implementace se nachází v souborech:

`file: patternfsel.m` - selekce příznaků (odstraní z matice ty kde je ve selekčním vektoru 0)

`file: autofex.m` skript, který provede automatickou selekci a extrakci příznaků, uloží do souborů:

`patterns.mat` ALL_F_SPACE - vzory(patterns)

`selmmask.mat` selMMask - masky selekce příznaků

`patternsFAT.mat` FAT - file system vzorů (mapování kde jsou třídy v ALL_F_SPACE)

`Wij.mat` - W-extrakce po jednotlivé závady i, j

6) 7-dim fuzzy hyperelipsoidní klasifikátor

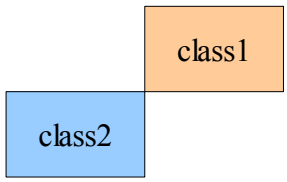
6.1 Proč jsem "zahodil" všechy klasifikátory v PRTools.

Tvrzení 6.1. Klasifikátor má úlohu znalostního systému - paměti, která zaznamená vzory.

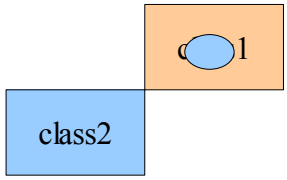
Z toho vyplývá požadavek na klasifikátor: V případě, že na vstupy klasifikátoru jsou přiložena data, která klasifikátor nikdy neobdržel, klasifikátor musí svůj výstup nastavit do stavu "nevím". Nikdy si nesmí bez zásahu inženýra vymýšlet diagnózy, které nejsou podloženy žádným statistickým odhadem. V případě známých pásem příznakového prostoru může inženýr klasifikátoru dodefinovat známé oblasti.

Bohužel ani jeden klasifikátor v PRTools neodpovídá tomuto modelu. Testoval jsem jak statistické tak neuronové sítě. Základní problém všech je, že dvě třídy se snaží rozdělit nadplochou, ale v oblastech bez výskytu dat si "domyslí" do které třídy prvky zařadí.

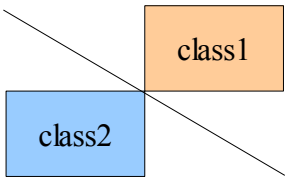
Naprosto tragicky dopadl klasifikátor RBF- neuronová síť s rotačně symetrickou jádrovou funkcí, který dokázal v testech na normálním rozdělení dat (60bodů v každé třídě):



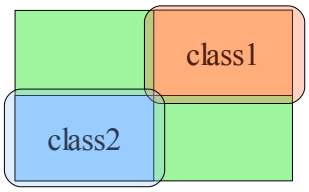
vygenerovat takovéto dělící nadplochy, navíc se chovala nahodile:



Bayes, k-nearest dopadl nejlépe, i když taky nevyhovujíce:



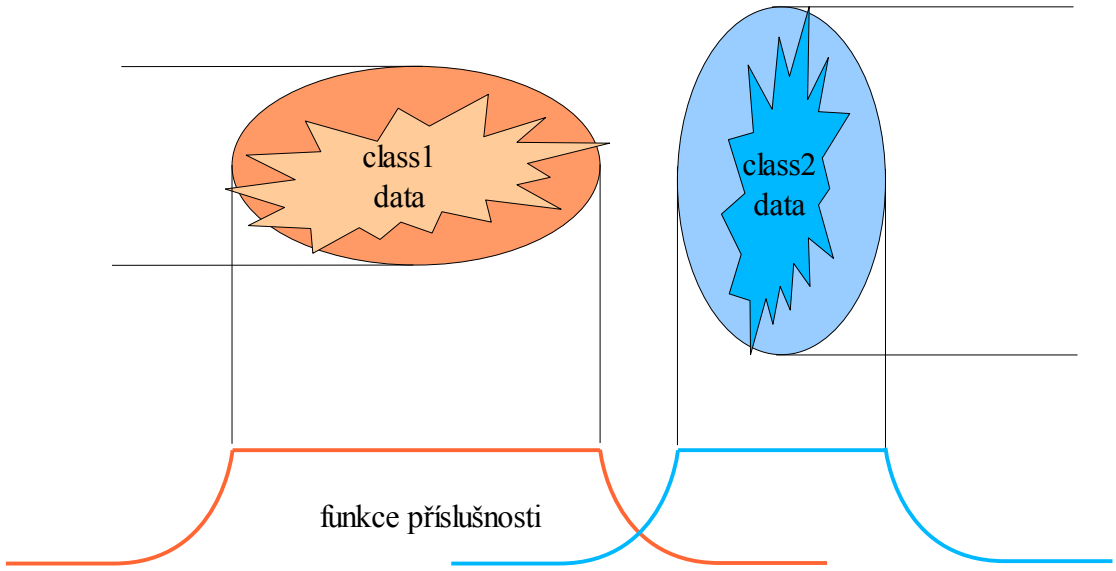
Jediný správný odhad je takovýto:



Zelená oblast je neznámá a klasifikátor nesmí v takovém případě zařadit do class1 nebo class2, ale do class_unknown. Class_unknown v případě expertní znalosti dodefinuje inženýr.
Jak dobře se odhadnou dělící nadplochy tříd je dáno použitou statistickou metodou. Zdůrazňuji statistickou, jelikož pokud nejsou data, a neznáme pravděpodobnostní rozdělení, nelze udělat nic jiného, než odhadnout dostatečně veliký rozptyl.

6.1 (fuzzy) hyper-elipsoidní klasifikátor

Elipsoidní - odhaduje se rozptyl v každé dimenzi, ten pak určuje škálování dimenzí.
Hyper - vícedimenzionální, v implementaci *W* extrakce redukuje na 7 dimenzí
Fuzzy - množina elipsoidu není ostře definována. Pokud bod padne do elipsoidu, příslušnost do shluku třídy (množiny) je 1. V případě, že je dále, klesá příslušnost s funkcí 1/R. R je poloměr od středí hodnoty.
Nefuzzy verze vrací 1/0 podle toho zda prvek je v elipsoidu (1) nebo ne (0).



Takováto reprezentace vkládá do znalostních vzorů neurčitost - vágnost - ta je všudy-přítomná, jelikož jsme omezeni množstvím dat a rozptyl není dostatečně kvalifikovaný odhad. Dále tento klasifikátor deformuje množiny na elipsoidy, což v některých případech může být na závadu. Z časových důvodů jsem implementoval to nejednodušší.

Implementace se nachází v souborech:

`trainhyperellip.m` - výpočet elipsoidu ze zadaných vzorů
`isinhyperellip.m` - výpočet příslušnosti bodu v elipsoidu
`trainec.m` skript, který provede automatické "naučení" klasifikátoru

vyžaduje (z **`autofex.m`**):
`patterns.mat` - vzory(patterns)
`patternsFAT.mat` - FAT vzorů
`selmmask.mat` - masky selekce příznaků

uloží natrénovaný klasifikátor do: `hyperellip_classifier.mat`

7) Diagnostický systém - `puzuk.m`

7.1 Odkud název Puzuk

Název je převzán z divadelní hry Ztížená možnost soustředění, autor V. Havel. Puzuk zde vystupoval jako expertní systém pro automatickou analýzu lidí, který příliš nefungoval z hardwarových důvodů :-).

7.2 Vstupy

`selmmask.mat` - selekce příznaků
`Wij.mat` - jednotlivé extrakce pro vady i, j (W23,W24..W29,W34..W39, ..., ...W89)
`hyperellip_classifier.mat` - natrénovaný klasifikátor
`t.mat` - změřená data
`classify.m` - funkce která hodnotí příslušnost vektoru příznaků do třídy

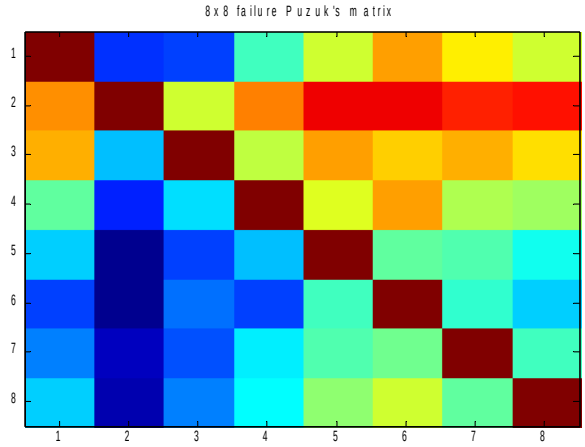
7.3 Výstup

Koincidenční matice 8x8 určující vazby mezi vadami. Na řádku *r*, soupci *c* (tj. vada *r* vůči vadě *c*) je uvedena hodnota klasifikátoru.

- Pokud je číslo kladné, jde spíše o vadu označenou řádkem.
- Pokud je číslo záporné, jde spíše o vadu označenou sloupcem.

Z uvedeného je zřejmé, že matice je "znaménkově[1]" diagonálně symetrická.

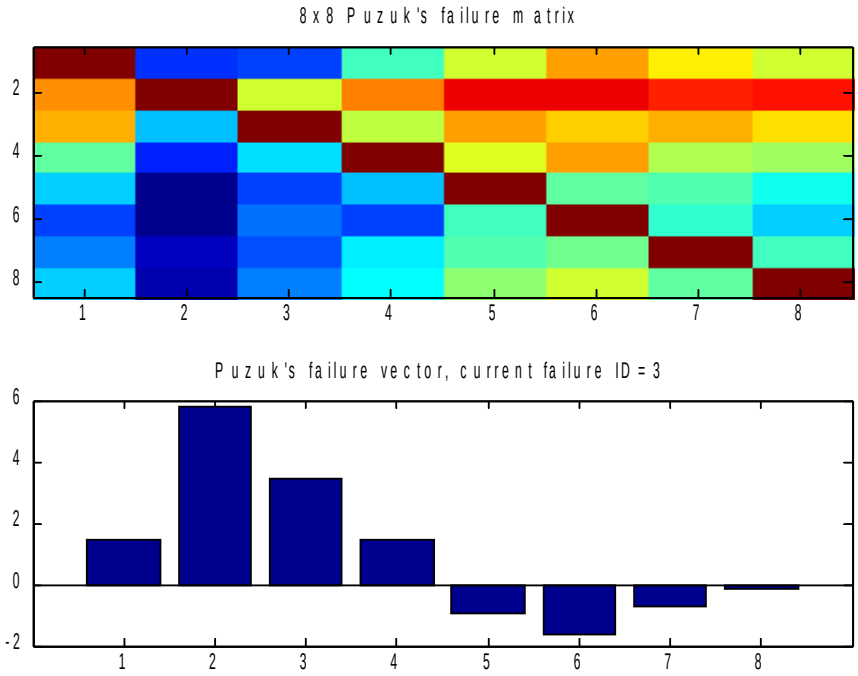
Příklad. (barva: odstíny modré -, odstíny červené +, zelená 0, hnědá=+1.1)



Součet sloupců (zleva doprava) pak určí vektor, kde index nejvyššího prvku je číslo vady.
Nebo: Součet řádků (zhora dolu) pak určí vektor, kde index nejnižšího prvku je číslo vady.

1 znaménkově je myšleno - až na znaménko je diagonálně symetrická

Výstup:



Poznámka. Failure ID je o 1 větší, protože jsou chyby číslo 2 až 9.

8) Confusion matrix

8.1 Data bez uvedení "nevím" - tj. i se špatnou klasifikací - fuzzyfikovaná verze, rozptyl*3. `(file autocfm.m)`

cca 30% dat nahodile vybráno pro trénování, 100% testovacích:
4% chybně zařazených
FilesPerFailure=[0 1 7 6 2 2 3 4 3];

	f2	*f3*	*f4*	*f5*	*f6*	*f7*	*f8*	*f9*
f2	2							
f3		31	5					
f4			30					
f5				8				
f6					8			
f7						16		
f8							18	
f9								16

legenda: kde je prázdné místo je 0.
cca 50% dat nahodile vybráno pro trénování, 100% testovacích:

5,6% chybně zařazených
FilesPerFailure=[0 1 9 8 2 2 4 4 4];

	f2	*f3*	*f4*	*f5*	*f6*	*f7*	*f8*	*f9*
f2	4							
f3		66	5	1				
f4		7	53					
f5		2		14				
f6					16			
f7						32		
f8							36	
f9								32

legenda: kde je prázdné místo je 0.

cca 75% dat nahodile vybráno pro trénování, 100% testovacích:
4% chybně zařazených
FilesPerFailure=[0 1 13 12 3 3 6 6 6];

	f2	*f3*	*f4*	*f5*	*f6*	*f7*	*f8*	*f9*
f2	4							
f3		63	9	1				
f4			60					
f5		1		15				
f6					16			
f7						32		
f8							36	
f9								32

– nefuzzyfikovaná verze na další straně -

8.2 Data s uvedením "nevím" - nefuzzyfikovaná verze s užším rozptylem *2.

cca 50% dat nahodile vybráno pro trénování, 100% testovacích:
0,4% chybně zařazených
5% nezařazených
FilesPerFailure=[0 1 9 8 2 2 4 4 4];

Nezařazeno (nevím) = 13

	f2	*f3*	*f4*	*f5*	*f6*	*f7*	*f8*	*f9*
f2	4							
f3		65	1					
f4			60					
f5				14				
f6					16			
f7						30		
f8							34	
f9								31

legenda: kde je prázdné místo je 0.

8.3 Zhodnocení kvalifikace

Rozdíl ve schopnostech diagnostikovat problém mezi klasifikátorem s fuzzy množinami a "nefuzzy" je ten, že fuzzy množiny se snaží odhadovat situaci podle znalosti (polohy a rozměrů hyperelipsoidu), kdežto "nefuzzy" se drží striktně statistických znalostí.

Konstruktér musí rozhodnout kde je možné dovolit diagnostice nechat dělat chyby (např. špatné rozhodnutí mezi chybou 3 a 4, nebo mezi chybou 3 a 9, 9 je totiž stav bez vady a znamená to, že hardware je považován za bezvadný!

9) Komentáře ke stávajícímu řešení

9.1 obecný komentář k diagnostice

- Veškerý úspěch diagnostiky leží na výběru příznakového prostoru a je nesmyslné vymýšlet "převelegeniální[2]" klasifikátory. Klasifikátor má roli paměti vzorů a měl by být založen na statistickém odhadu a dobré aproximaci množiny. Domnívám se, že dostačující typ by byl založen na shlukové analýze, nahrazení lokálních shluků hyperelipsoidy. Tak se dají modelovat všechny množiny, které jsou dobře statisticky podložené (lokální shluk by měl tvořit minimálně 30 vzorů).

2 newspeak, z 1984, George Orwell

- Reálné lineární *W*-extrakce lze používat pouze za platnosti lineární separovatelnosti multimnožin viz trvzení 5.1. Je nutné najít nebo vymyslet metodu která generuje *W* a zároveň neporušuje trvzení 5.1.
- Metody, které je pořeba rozvíjet je analýza prostoru příznaků. Separace založená na analýze informační kvality příznaků (např. zmíněná metoda minimalizace konjunkce množin). V začátku designu je třeba vyzkoušet všelijaké příznaky - např. použil jsem RMS, spektrum, polynomiální aproximaci spektra, FFT transformaci spektra (tj. FFT(log(segmentation(FFT(signal)))), zbývá crest faktor, max amplita atd... Užitečná se při selekcích trochu jevila FFT(spektra), při selekci nebyla zahozena. Základní problém je v nekonečné řadě lineárních transformací. Výhodou je, že všechy reálné lineární *W*-transformace snižující dimenzi lze hodit do koše pokud neplatí tvrzení 5.1. S komplexními transformacemi nevím jak se to má, ale bude tam taky nějaké kritérium:-) .
- Jak realizovat takové prohledávání:
  - 1. inženýr definuje pokud "vidí okem"
  - 2. na základní vstupní signál lze aplikovat alogoritmus, který v signálu hledá "primitiva", taková primitiva mohou být polynomy, bázové funkce transformací, geometrické útvary... Tato nalezená primitiva pak slouží jako "podpis" signálu. Primitiva tvoří "báze prostoru". Nevědomky je přesně tento přístup FFT, polyfit, wavelet transformace. Na hledání primitiv lze pak použít principy genetického programování, pro které je fitness funkce právě kritérium separability tříd, nebo vyzkoušet vše hrubou silou - co se dá zvládnout v dostupném čase.

9.2 omezení stávajícího návrhu a vylepšení

- spektra FFT dobře vystihují stacionární signál a jeho "zvlněnost" v časové doméně
- pomocí extrakce příznaků přes *W* "nejsme" schopni určit číselně "zvlněnost" spektra (pokud *W* není sama FFT transformací[3], tj. je dostatečně jednoduchá:-). Důvod proč o tom mluvím je, že např. vada může způsobit nepatrný útlum nějakých složek a jiné velmi málo podpořit. Jestliže budeme zkoumat velikost jednotlivých amplitud, je to obdobé jako kdybychom se snažili stanovit spektrum v časové oblasti. V tomoto případě aplikací wavletové nebo fourierovy transformace na spektrum tak získáme povahu spektra ve "spektru" vlnek a údaje skryté ve spoustě čísel tak získáme pár čísly.
- testoval jsem i polynomiální proklad spektra `polyfit`, viz. "reliktní" proměnná SpolyN ve skriptu autofex.m, selekce příznaků je ve většině případů naprosto vyloučila.
- **Co tento model postihuje:** vada může být zdrojem neharmonických vibrací, a to tyto vibrace mohou stimulovat rezonanční módy celého systému. Celkově se to pak projeví na zvýraznění peaků ve spektru tj. spektrum začne být velice zvlněné. Jednotlivé peaky, jelikož módů je ohromné množství, nepředstavují přínosnou informaci. Jde

3 uvažujeme, že je *W* reálná, pro FFT je komplexní

hlavě o efekt většího zvlnění spektra, ale na jakých frekvencích, to nelze pořádně odhadnout.