Univerza v Ljubljani, Fakulteta za elektrotehniko
Laboratorij za slikovne tehnologije

Biomedicinska informatika

LABORATORIJSKE VAJE

Tomaž Vrtovec

2012

Univerza v Ljubljani, Fakulteta za elektrotehniko
Laboratorij za slikovne tehnologije

Biomedicinska informatika

LABORATORIJSKE VAJE

Tomaž Vrtovec

2012

Kazalo

Univerza v Ljubljani, Fakulteta za elektrotehniko
Podiplomski študijski program 2. stopnje Elektrotehnika
Biomedicinska tehnika - Biomedicinska informatika (64209) - Laboratorijske vaje

Vaja 1: Uvod v Matlab

Navodila

Uvodna vaja služi spoznavanju ravnanja s spremenljivkami ter uporabe osnovnih ukazov in funkcij v programskem okolju Matlab. V ta namen razvijte algoritem za urejanje naključnih števil (po velikosti) po principu mehurčnega urejanja (*ang.* bubble sort).

1. S pomočjo funkcije `randperm()` določite vektor neurejenih števil, pri čemer vhodni argument N predstavlja število elementov v vektorju, ki zavzemajo celostevilčne vrednosti med 1 in N, izhodni argument pa vrstični vektor neurejenih števil.

2. Napišite funkcijo za urejanje naključnih števil po principu mehurčnega urejanja:

   ```
   function oVector = bubbleSort(iVector),
   ```

   kjer je vhodni argument `iVector` vektor neurejenih števil, izhodni argument `oVector` vektor urejenih števil. Oba vektorja sta seveda enakih velikosti.

3. Napišite funkcijo za prikazovanje dobljenih rezultatov:

   ```
   function displayResults(iVector, iColor, iName),
   ```

   kjer je vhodni argument `iVector` poljuben vektor števil, ki jih želite prikazati, `iColor` je barva izrisa (npr. 'r' za rdečo, 'b' za modro, itn.), `iName` pa je naslov prikaznega okna oz. grafa. Uporabite funkcijo `plot()`, prikaz pa poljubno prilagodite s pomočjo ukazov `axis` in `grid` ter funkcij `title()`, `xlabel()` in `ylabel()`.

4. Preizkusite delovanje algoritma za mehurčno urejanje na podlagi velikosti vektorjev $N = 50$ in $N = 1000$.

Vprašanja

Poročilo vaje vsebuje lastnoročno (brez uporabe računalnika) zapisane odgovore na naslednja vprašanja, medtem ko so zahtevane slike natisnjene ter priložene poročilu.

1. Priložite sliko, ki prikazuje položaj elementa v vektorju v odvisnosti od njegove vrednosti pred in po urejanju po velikosti, in sicer za velikost vektorja $N = 50$.

2. Priložite sliko, ki prikazuje položaj elementa v vektorju v odvisnosti od njegove vrednosti pred in po urejanju po velikosti, in sicer za velikost vektorja $N = 1000$.

Vaja 2: Elektrokardiogram (EKG)

Navodila

Elektrokardiografija – EKG (*ang.* electrocardiography – ECG) je neinvazivno merjenje električne aktivnosti srca vzdolž časa preko elektrod, ki so pritrjene na površino kože. Grafična predstavitev meritev se imenuje elektrokardiogram (tudi EKG), podatke pa sestavljajo tipični periodični signali, ki opisujejo srčni utrip, sestavljeni so pa iz P vala, QRS sestava (Q val, R val in S val) in T vala ter iz vmesnih PQ in ST segmentov. Iz spletne zbirke EKG podatkov boste pridobili tipične EKG podatke ter na podlagi analize v časovnem prostoru določili nekatere osnovne parametre srčnega utripa.

1. Na spletni strani PhysioNet (http://www.physionet.org) izberite "PhysioBank" → "Signal Archives" → "ECG" ter nato zbirko EKG podatkov "European ST-T Database". Nato shranite izbrane EKG podatke, in sicer datoteko z opisom (*.hea) ter datoteko s podatki (*.dat). S pomočjo programa ecg2mat.m pretvorite shranjene EKG podatke iz formata 212 (*.dat) v Matlabov binarni format (*.mat), pri čemer čas začetka opazovanja nastavite na 0 s, čas konca opazovanja pa na 5 s.

2. Pridobljene binarne podatke naložite v okolje Matlab s pomočjo funkcije `load()`. Naloženi podatki predstavljajo strukturo (npr. `ecg`), v kateri je opis obeh EKG signalov (`ecg.desc1` in `ecg.desc2`), podatki obeh signalov (`ecg.signal1` in `ecg.signal2`) ter pripadajoči čas (`ecg.time`), pri katerem so bili vzorci signala zajeti. Izberite signal, ki predstavlja 4. prsni odvod (V_4), ter ga prikažite na zaslon.

3. Napišite funkcijo za iskanje vrhov Q vala, R vala in S vala v QRS sestavu:

   ```
   function [oIdxQ, oIdxR, oIdxS] = detectWavePeaksQRS(iData),
   ```

   kjer vhodni argument `iData` predstavlja vektor vzorcev podatkov, izhodni argumenti `oIdxQ`, `oIdxR` in `oIdxS` pa vektorje položajev (indeksov) vrhov Q vala, R vala in S vala v vektorju vzorcev podatkov. Upoštevajte, da R val nastopi kot maksimum na območju, kjer so vrednosti večje od 50% razpona EKG podatkov, Q val in S val pa predstavljata prvi minimum na območju levo in desno od R vala. Dobljene položaje in pripadajoče vrednosti vrhov valov izrišite na zaslon v isto sliko kot EKG signal.

4. Napišite funkcijo za iskanje vrhov P in T vala:

   ```
   function [oIdxP, oIdxT] = detectWavePeakPT(iData, iIdxQ, iIdxS),
   ```

   kjer vhodni argument `iData` predstavlja vektor vzorcev podatkov, `iIdxQ` in `iIdxS` pa položaje (indekse) vrhov Q vala in S vala v vektorju vzorcev podatkov. Izhodna argumenta `oIdxP` in `oIdxT` predstavlja položaje (indekse) vrhov P vala in T vala v vektorju vzorcev podatkov. Upoštevajte, da P val nastopi levo od Q vala na položaju, ki je od njega oddaljen kvečjemu za tretjino razdalje med vrhom Q vala in vrhom predhodnega S vala. Podobno upoštevajte, da T val nastopi desno od S vala na položaju, ki je od njega oddaljen kvečjemu za tretjino razdalje med vrhom S vala in vrhom naslednjega Q vala. Dobljene

položaje in pripadajoče vrednosti vrhov valov izrišite na zaslon v isto sliko kot EKG signal.

5. Napišite funkcijo za določanje srčnega utripa na podlagi izbranega vala v signalu:

   ```
   function [oHB_avg, oHB_std, oHF_avg, oHF_std] = computeHeartBeat(iTime, iIdx),
   ```

   kjer vhodni argument `iTime` predstavlja vektor časa, pri katerem so bili vzorci podatkov zajeti, `iIdx` pa predstavlja položaje (indekse) vrhov izbranega vala v vektorju vzorcev podatkov. Izhodna argumenta `oHB_avg` in `oHB_std` predstavljata povprečno periodo in pripadajoči standardni odklon pojavljanja vrha izbranega vala (v sekundah), `oHF_avg` in `oHF_std` pa povprečno frekvenco in pripadajoči standardni odklon pojavljanja vrha izbranega vala (v hertzih).

Vprašanja

Poročilo vaje vsebuje lastnoročno (brez uporabe računalnika) zapisane odgovore na naslednja vprašanja, medtem ko so zahtevane slike natisnjene ter priložene poročilu.

1. Čas začetka opazovanja EKG signala nastavite na 0 s, čas konca opazovanja pa prvič na 5 s, drugič na 50 s in tretjič na 500 s. Za vsak eksperiment priložite sliko izrisa EKG signala ter vrhov Q, R in S vala v QRS sestavu ter vrhov P in T vala.

2. Za vsak eksperiment zapišite povprečne frekvence pojavljanja vrha vala in pripadajoče standardne odklone (enote naj bodo udarci na minuto), in sicer za vsak val posebej. Frekvenca pojavljanja vrhov katerega vala je najbolj spremenljiva, na podlagi česa ste to ugotovili in kaj je vzrok spremenljivosti?

3. Zakaj je velikost datoteke s podatki (`*.dat`) natančno 5.400.000 bajtov?

Dodatek

Odgovore na sledeče probleme ni potrebno prilagati poročilu. Služijo naj v razmislek ter kot vzpodbuda za boljše razumevanje vsebine.

Priložite sliko izbranega EKG signala, in sicer prikazanega na "navideznem" milimeterskem EKG papirju, pri čemer upoštevate standardne velikosti razdelkov. Razmerje osi koordinatnega sistema nastavite z Matlabovo funkcijo `dascpect([dx dy 1])`, kjer `dx` in `dy` predstavljata velikost posameznega razdelka v x in y smeri. Zaradi smiselnosti prikaza nastavite čas začetka opazovanja na 0 s, čas konca opazovanja pa na 5 s.

Vaja 3: Standard DICOM

Navodila

Standard DICOM za digitalne slikovne tehnologije in komunikacijo v medicini (*ang.* Digital Imaging and Communications in Medicine) je uveljavljen tako med proizvajalci medicinskih slikovnih naprav kot tudi v zdravstvenih organizacijah za shranjevanje in pregledovanje medicinskih slik.

Dana je DICOM datoteka imageXY.dcm, ki vsebuje sliko, zajeto s slikovno tehniko magnetne resonance (MR). Na podlagi DICOM standarda izluščite željene podatkovne elemente iz datoteke. Za določanje podatkovnih elementov so vam v pomoč skupine oznak v datoteki dcmGroups.mat, elementi oznak v datoteki dcmElements.mat in predstavitve oznak v datoteki dcmRepresentations.mat. Datoteke lahko naložite s pomočjo Matlabove funkcije `load()`, vsaka naložena struktura `S` pa vsebuje spremenljivko `S.code` s kodo in spremenljivko `S.desc` z opisom.

1. Napišite funkcijo za nalaganje DICOM datoteke:

   ```
   function oData = loadDicomFile(iPath),
   ```

   kjer vhodni argument `iPath` predstavlja pot (mapa ter ime datoteke) do DICOM datoteke. Izhodni argument `oData` predstavlja podatke v DICOM datoteki v obliki vrstičnega vektorja za vsak bajt podatkov posebej. Podatke preberete s pomočjo Matlabovih funkcij `fopen()`, `fread()` in `fclose()`, pri čemer upoštevate vrsto podatkov `'uint8'` (nepredznačena 8-bitna števila).

2. Napišite funkcijo za pridobivanje podatkovnega elementa iz podatkov v DICOM datoteki:

   ```
   function oElement = getDicomDataElement(iData, iTag),
   ```

   kjer vhodni argument `iData` predstavlja podatke v DICOM datoteki, `iTag` pa oznako podatkovnega elementa oblike (skupina,element) = (ssss,eeee). Izhodni argument `oElement` je v obliki strukture, ki vsebuje naslednje spremenljivke, povezane s podatkovnim elementom:

   - `oElement.idx` je položaj (*ang.* index) začetka zapisa podatkovnega elementa v vektorju DICOM podatkov, npr. `585`.
   - `oElement.tag` je oznaka (*ang.* tag) podatkovnega elementa oblike (skupina,element) = (ssss,eeee), npr. `'0008,0023'`.
   - `oElement.VR` je predstavitev vrednosti (*ang.* value representation, VR) podatkovnega elementa, npr. `'DA'` (predstavlja datum).
   - `oElement.VL` je dolžina vrednosti (*ang.* value length, VL) podatkovnega elementa v bajtih, npr. 8.
   - `oElement.value` je vrednost (*ang.* value) podatkovnega elementa, npr. `'20120516'`.

   Za pretvorbo števil iz desetiškega v šestnajstiški oz. šestnajstiškega v desetiški številski sistem si lahko pomagate z Matlabovo funkcijo `dec2hex()` oz. `hex2dec()` (npr. `dec2hex(35) = '23'` in `dec2hex(35, 4) = '0023'` oz. `hex2dec('23') = 35` in `hex2dec('0023') = 35`).

   Za pretvorbo številčnih vrednosti v besedilo si lahko pomagate z Matlabovo funkcijo `char()` (npr. `char(68) = 'D'` in `char([68, 65]) = 'DA'`).

3. Prilagodite izhodni argument `oElement.value` funkcije `getDicomDataElement()` tako, da bo ustrezal predstavitvi vrednosti, podani z `oElement.VR`. V ta namen napišite funkcijo:

```
function oValue = convertDicomValue(iValue, iVR),
```

   kjer vhodni argument `iValue` predstavlja vrednost, `iVR` pa kodo predstavitve vrednosti podatkovnega elementa. Izhodni argument `oValue` predstavlja ustrezno prilagojeno (pretvorjeno) vrednost podatkovnega elementa. Za predstavitve vrednosti upoštevajte kode AS, CS, DA, DS, LO, PN, SH, ST, TM, UL in US.

   Primer: Če je koda predstavitve enaka `oElement.VR = 'DA'` in vrednost enaka `oElement.value = [50, 48, 49, 50, 48, 53, 49, 54]`, potem je ustrezna pretvorba s pomočjo Matlabove funkcije `char()` enaka datumu (koda DA) `oElement.value = '20120516'` oblike LLLLMMDD.

4. Sliko prikažite s pomočjo priložene funkcije `displayDicomImage(iData)`, kjer vhodni argument `iData` predstavlja podatke v DICOM datoteki.

Vprašanja

Poročilo vaje vsebuje lastnoročno (brez uporabe računalnika) zapisane odgovore na naslednja vprašanja, medtem ko so zahtevane slike natisnjene ter priložene poročilu.

1. Zapišite ime izbrane DICOM datoteke, iz katere ste izluščevali podatkovne elemente.

2. Pridobite podatkovne elemente iz skupine informacij o preiskavi (*ang.* Study Information), in sicer zapišite vrednosti za datum slikanja (*ang.* Image Date), vrsto slike (*ang.* Modality), proizvajalca slikovne naprave (*ang.* Manufacturer) ter inštitucijo, v kateri je bilo slikanje opravljeno (*ang.* Institution Name). Pri vsaki vrednosti podajte tudi položaj (indeks) začetka podatkovnega elementa v vektorju DICOM podatkov, oznako oblike (ssss,eeee), predstavitev vrednosti (VR) in dolžino vrednosti (VL, v bajtih).

3. Pridobite podatkovne elemente iz skupine informacij o bolniku (*ang.* Patient Information), in sicer zapišite vrednosti za ime bolnika (*ang.* Patient's Name), datum rojstva bolnika (*ang.* Patient's Birth Date), spol bolnika (*ang.* Patient's Sex), starost bolnika (*ang.* Patient's Age) ter težo bolnika (*ang.* Patient's Weight). Pri vsaki vrednosti podajte tudi položaj (indeks) začetka podatkovnega elementa v vektorju DICOM podatkov, oznako oblike (ssss,eeee), predstavitev vrednosti (VR) in dolžino vrednosti (VL, v bajtih).

4. Pridobite podatkovne elemente iz skupine infomacij o slikah (*ang.* Image Information), in sicer zapišite vrednosti za število vrstic (*ang.* Rows), število stolpcev (*ang.* Columns), število rezerviranih bitov (*ang.* Bits Allocated), središče okna (*ang.* Window Center) ter širino okna (*ang.* Window Width). Pri vsaki vrednosti podajte tudi položaj (indeks) začetka podatkovnega elementa v vektorju DICOM podatkov, oznako oblike (ssss,eeee), predstavitev vrednosti (VR) in dolžino vrednosti (VL, v bajtih).

5. Priložite sliko, ki je shranjena v izbrani dani DICOM datoteki.

6. Koliko je dolžina (v bajtih) ... :

   - ... izbrane DICOM datoteke?
   - ... slikovnih podatkov?
   - ... vseh podatkov razen slikovnih?
   - ... vseh podatkovnih elementov?

   Odgovore ustrezno obrazložite.

7. Izpišite besedilo, ki je v izbrani DICOM datoteki shranjeno na položaju med bajtom 129 in bajtom 132. Na kaj sklepate glede na dobljeno besedilo?

Dodatek

Odgovore na sledeče probleme ni potrebno prilagati poročilu. Služijo naj v razmislek ter kot vzpodbuda za boljše razumevanje vsebine.

Prilagodite izhodni argument `oElement` funkcije `getDicomDataElement()` tako, da določite še:

- `oElement.group` je ime skupine oznak, ki jih pridobite iz datoteke dcmGroups.mat. Dobljena struktura določa skupine, npr. na položaju 6 je `G.code{6} = '0028'` in pripadajoči opis je `G.desc{6} = 'Image Information'`.
- `oElement.element` je ime elementa oznak, ki jih pridobite iz datoteke dcmElements.mat. Dobljena struktura določa elemente, npr. na položaju 96 je `E.code{96} = '0028,0100'` in pripadajoči opis je `E.desc{96} = 'Bits Allocated'`.

Vaja 4: Šifriranje podatkov

Navodila

Šifriranje podatkov ali kriptografija je znanstvena veda o skrivnem pisanju sporočil in njihovemu razkrivanju, omogoča pa mehanizme za zaščito, zasebnost in zaupnost podatkov. Na podlagi preprostega algoritma in skrivnega ključa šifrirajte in dešifrirajte dano besedilo, pri čemer ga v uporabno obliko pretvorite s kodiranjem po shemi Base64.

1. Napišite funkcijo za šifriranje besedila na podlagi skrivnega ključa:

   ```
   function oText = encryptText(iText, iKey),
   ```

   kjer vhodni argument `iText` predstavlja čistopis, `iKey` pa šifrirni ključ. Izhodni argument `oText` predstavlja šifropis. Šifriranje opravite tako, da številčne vrednosti čistopisa prištejete številčnim vrednostim ključa, pri čemer ključ po potrebi ponavljate do dolžine čistopisa. Šifropis naj bo zakodiran po shemi Base64, v ta namen pa napišite funkcijo:

   ```
   function oText = encodeBase64(iText),
   ```

   kjer vhodni argument `iText` predstavlja začetno besedilo, izhodni argument `oText` pa besedilo, zakodirano po shemi Base64.

   Za določanje številčnih vrednosti na podlagi znakov besedila uporabite Matlabovo funkcijo `unicode2native()`, za določanje znakov besedila na podlagi številčnih vrednosti pa uporabite Matlabovo funkcijo `native2unicode()`. Za pretvorbo med desetiškim in dvojiškim sistemom uporabite Matlabovi funkciji `dec2bin()` in `bin2dec()`. Številčne vrednosti in znake besedila za kodiranje po shemi Base64 pridobite iz datoteke tableBASE64.mat (npr. struktura `BASE64`), v kateri spremenljivka `BASE64.charCode` predstavlja številčne vrednosti, spremenljivka `BASE64.charSymbol` pa pripadajoče znake (npr. `BASE64.charCode(13) = 12` in `BASE64.charSymbol(13) = 'M'`.

2. Napišite funkcijo za dešifriranje besedila na podlagi skrivnega ključa:

   ```
   function oText = decryptText(iText, iKey),
   ```

   kjer vhodni argument `iText` predstavlja šifropis, `iKey` pa dešifrirni ključ. Izhodni argument `oText` predstavlja čistopis. Dešifriranje opravite tako, da številčne vrednosti ključa odštejete od številčnih vrednosti šifropisa, pri čemer ključ po potrebi ponavljate do dolžine šifropisa. Šifropis naj bo najprej dekodiran po shemi Base64, pri čemer zaključne bajte predstavlja znak '=', v ta namen pa napišite funkcijo:

   ```
   function oText = decodeBase64(iText),
   ```

   kjer vhodni argument `iText` predstavlja začetno besedilo, izhodni argument `oText` pa besedilo, dekodirano po shemi Base64.

   Postopek je obraten šifriranju besedila, pomagajte pa si z enakimi Matlabovimi funkcijami, kot je opisano pod točko 1.

3. Napišite funkcijo za nalaganje besedila iz tekstovne datoteke:

   ```
   function oText = loadText(iPath),
   ```

   kjer vhodni argument `iPath` predstavlja pot (mapa ter ime datoteke) do tekstovne datoteke, izhodni argument `oText` pa besedilo v obliki vrstičnega vektorja.

   Podatke preberete s pomočjo Matlabovih funkcij `fopen()`, `fread()` in `fclose()`, pri čemer upoštevate vrsto podatkov 'uint8' (nepredznačena 8-bitna števila), v besedilo pa jih pretvorite s pomočjo Matlabove funkcije `native2unicode()`.

Vprašanja

Poročilo vaje vsebuje lastnoročno (brez uporabe računalnika) zapisane odgovore na naslednja vprašanja, medtem ko so zahtevane slike natisnjene ter priložene poročilu.

1. Naložite čistopis iz datoteke plainText.txt ter ga najprej šifrirajte z opisanim algoritmom, pri čemer za skrivni ključ uporabite niz 'enkripcija', in nato zakodirajte po shemi Base64. Zapišite čistopis ter dobljeni šifropis.

2. Naložite šifropis iz datoteke cihperText.txt ter ga najprej dekodirajte po shemi Base64 in nato dešifrirajte z opisanim algoritmom, pri čemer za skrivni ključ uporabite niz 'dekripcija'. Zapišite šifropis ter dobljeni čistopis.

3. Koliko je dolžina (v bajtih) ...:

   - ... čistopisa v datoteki plainText.txt?
   - ... šifriranega in zakodiranega čistopisa?
   - ... šifropisa v datoteki cipherText.txt?
   - ... dekodiranega in dešifriranega šifropisa?

   Odgovore ustrezno obrazložite v povezavi z dolžino začetnega besedila (čistopisa oz. šifropisa).

4. Zakaj je uporabno kodiranje po shemi Base64? Obrazložite odgovor.

Dodatek

Odgovore na sledeče probleme ni potrebno prilagati poročilu. Služijo naj v razmislek ter kot vzpodbuda za boljše razumevanje vsebine.

Datoteka secretText.txt vsebuje šifropis, ki je bil šifriran s pomočjo predstavljenega algoritma na podlagi neznanega skrivnega ključa. Na osnovi izčrpnega iskanja (preverjanja vseh možnih kombinacij) dešifrirajte besedilo ter poiščite skrivni ključ, pri čemer upoštevate informacije, da besedilo vsebuje besedo "Janez" ter da je skrivni ključ dolg 3 znake in vsebuje samo male ali velike črke. Za iskanje prisotnosti niza znakov v danem besedilu si pomagajte z Matlabovo funkcijo `strfind()`.

Vaja 5: Binarno razvrščanje

Navodila

Raziskovalci so za namene preverjanja predpostavke, da je ena izmed bolezni srca in ožilja povezana z sistoličnim (višjim) krvnim tlakom, razvili štiri različne diagnostične teste (metode) merjenja krvnega tlaka. Vsi testi merijo krvni tlak v območju med 70 mmHg in 170 mmHg (nižjih ali višjih vrednosti ne zaznajo), z njimi pa so izmerili krvni tlak pri 2268 osebah, od katerih je bila ena polovica bolnih (1134 oseb), druga polovica pa zdravih (1134 oseb). Rezultate so zapisali v datoteko labData.mat (npr. struktura `D`), v kateri spremenljivka `D.refData` predstavlja referenčne podatke (vrednost 0 predstavlja zdravo, vrednost 1 pa bolno osebo), spremenljivka `D.testData` pa predstavlja rezultate diagnostičnih testov (krvni tlak v mmHg), pri čemer vsaka vrstica `i` predstavlja posamezno osebo, vsak stolpec `j = 1...4` pa posamezen diagnostični test:

	D.refData	D.testData			
i	(i,1)	(i,1)	(i,2)	(i,3)	(i,4)
⋮	⋮	⋮	⋮	⋮	⋮
71	0	85.5	98.5	114.8	115.0
72	1	144.2	70.6	113.8	115.0
73	0	93.4	145.6	117.9	115.0
⋮	⋮	⋮	⋮	⋮	⋮

1. Napišite funkcijo za binarno razvrščanje rezultatov:

   ```
   function [oTP, oTN, oFP, oFN] = classifyData(iThreshold, iTestData, iRefData),
   ```

   kjer vhodni argument `iThreshold` predstavlja prag razvrščanja, `iTestData` je vektor rezultatov posameznega diagnostičnega testa, `iRefData` pa vektor referenčnih podatkov. Izhodni argumenti `oTP`, `oTN`, `oFP` in `oFN` predstavljajo število resnično pozitivnih (TP), resnično negativnih (TN), lažno pozitivnih (FP) in lažno negativnih (FN) rezultatov.

   Podatke razvrščajte glede na `tNum` različnih pragov, pri čemer jih enakomerno razporedite med najmanjšim pragom `tMin` = 70 mmHg in največjim pragom `tMax` = 170 mmHg.

2. Napišite funkcijo za računanje deležev, ki merijo sposobnost binarnega razvrščanja:

   ```
   function function [oTPR, oTNR, oFPR, oFNR] = computeRates(iTP, iTN, iFP, iFN),
   ```

   kjer vhodni argumenti `iTP`, `iTN`, `iFP` in `iFN` predstavljajo število resnično pozitivnih (TP), resnično negativnih (TN), lažno pozitivnih (FP) in lažno negativnih (FN) rezultatov. Izhodni argumenti `oTPR`, `oTNR`, `oFPR` in `oFNR` predstavljajo delež resnično pozitivnih rezultatov (TPR oz. občutljivost), delež resnično negativnih rezultatov (TNR oz. specifičnost), delež lažno pozitivnih rezultatov (FPR oz. nespecifičnost) in delež lažno negativnih rezultatov (FNR oz. neobčutljivost).

3. Napišite funkcijo za računanje preostalih vrednosti, ki ravno tako merijo sposobnost binarnega razvrščanja:

   ```
   function function [oPPV, oNPV, oFDR, oACC] = computeValues(iTP, iTN, iFP, iFN),
   ```

   kjer vhodni argumenti `iTP`, `iTN`, `iFP` in `iFN` predstavljajo število resnično pozitivnih (TP), resnično negativnih (TN), lažno pozitivnih (FP) in lažno negativnih (FN) rezultatov. Izhodni argumenti `oPPV`, `oNPV`, `oFDR` in `oACC` predstavljajo pozitivno napovedno vrednost (PPV), negativno napovedno vrednost (NPV), delež lažnega odkrivanja rezultatov (FDR) in točnost razvrščanja (ACC).

4. Napišite funkcijo za izris rezultatov razvrščanja:

   ```
   function drawResults(iX, iY, iTitle, iLabelX, iLabelY),
   ```

   kjer vhodna argumenta `iX` in `iY` predstavljata vrednosti na x in y osi izrisa, `iTitle` je naslov izrisa, `iLabelX` in `iLabelY` pa oznake na x in y osi izrisa.

   Vhodna argumenta `iX` in `iY` naj bosta matriki, pri čemer vsaka vrstica predstavlja posamezni prag razvrščanja, vsak stolpec pa posamezen diagnostični test. Rezultate razvrščanja vseh testov izrišete v isti koordinatni sistem, legendo z generičnimi oznakami (`data1`, `data2`, `data3`, `data4`) pa lahko izrišete s pomočjo Matlabove funkcije `legend('Location', 'BestOutside')`.

5. Napišite funkcijo za izračun površine pod krivuljo (AUC):

   ```
   function oAUC = computeAUC(iX, iY),
   ```

   kjer vhodna argumenta `iX` in `iY` predstavljata krivuljo oz. njene vrednosti na x in y osi izrisa, izhodni argument `oAUC` pa predstavlja površino pod krivuljo, ki jo izračunate po trapezni metodi.

Vprašanja

Poročilo vaje vsebuje lastnoročno (brez uporabe računalnika) zapisane odgovore na naslednja vprašanja, medtem ko so zahtevane slike natisnjene ter priložene poročilu.

1. Razvrstite rezultate na podlagi `tNum = 20` različnih pragov ter priložite slike poteka TPR, TNR, FPR in FNR v odvisnosti od praga razvrščanja.

2. Razvrstite rezultate na podlagi `tNum = 20` različnih pragov ter priložite slike poteka PPV, NPV, FDR in ACC v odvisnosti od praga razvrščanja.

3. Razvrstite rezultate na podlagi `tNum = 20` različnih pragov ter priložite slike ROC krivulj in podajte vrednosti površin pod ROC krivuljami. Na podlagi površine pod ROC krivuljo označite uspešnost posameznega diagnostičnega testa.

4. Na podlagi izrisanih potekov in krivulj določite optimalni prag razvrščanja za vsak diagnostični test posebej. Vašo izbiro ustrezno obrazložite.

5. Kateri diagnostični test je najboljši? Kaj lahko sklepate o testu, katerega ROC krivulja poteka popolnoma diagonalno? Kaj lahko sklepate o testu, katerega ROC krivulja poteka skoraj diagonalno? Odgovore ustrezno obrazložite.

6. Za prag razvrščanja izberite 120 mmHg in za vsak diagnostični test zapišite kontingenčno tabelo ter podajte deleže TPR, TNR, FPR in FNR, kot tudi vrednosti PPV, NPV, FDR in ACC.

Dodatek

Odgovore na sledeče probleme ni potrebno prilagati poročilu. Služijo naj v razmislek ter kot vzpodbuda za boljše razumevanje vsebine.

Napišite funkcijo za računanje praga razvrščanja, s katerim dosežemo izbrano občutljivost razvrščanja:

```
function [oThreshold, oTPR, oFPR] = computeThreshold(iTPR, iTestData, iRefData),
```

kjer vhodni argument `iTPR` predstavlja izbrani delež resnično pozitivnih rezultatov (TPR oz. občutljivost), `iTestData` je vektor rezultatov posameznega testa, `iRefData` pa vektor referenčnih podatkov. Izhodni argument `oThreshold` predstavlja prag, s katerim dosežemo izbrano občutljivost razvrščanja, `oTPR` predstavlja dejansko doseženo občutljivost razvrščanja, `oFPR` pa dejansko doseženo nespecifičnost razvrščanja.

Zapišite dobljene prage razvrščanja ter dejansko doseženo občutljivost in nespecifičnost razvrščanja pri izbranih občutljivostih razvrščanja 90.0%, 95.0% in 97.5%.

Vaja 6: Elektronski zdravstveni zapis

Navodila

Elektronski zdravstveni zapis - EZZ (*ang.* electronic health record, EHR) je koncept sistematičnega zbiranja zdravstvenih podatkov o posameznih bolnikih v elektronski obliki. Predpostavimo, da imamo EZZ zapisan z razširljivim označevalnim jezikom - XML (*ang.* extensible markup language), pri čemer je zapis nekoliko poenostavljen (značke ne vsebujejo atributov, ime značke se ne ponovi na istem nivoju, vrednost značke je vedno besedilo), npr. kot:

```
<medications>
      <medication2>
            <name>Ultop</name>
            <fullname>Ultop 10 mg trde kapsule</fullname>
            <usage>1 kapsula vsakih 24 ur</usage>
            <code>040762</code>
            <dateOfPrescription>
                  <dd>15</dd>
                  <mm>06</mm>
                  <yyyy>2012</yyyy>
            </dateOfPrescription>
            <dateOfExpiration>
                  <dd>06</dd>
                  <mm>07</mm>
                  <yyyy>2012</yyyy>
            </dateOfExpiration>
      </medication2>
</medications>
```

Zapis v obliki strukture, ki je enakovreden zgornjemu XML besedilu, je enak:

```
medications.medication2.name = 'Ultop'
medications.medication2.fullname = 'Ultop 10 mg trde kapsule'
medications.medication2.usage = '1 kapsula vsakih 24 ur'
medications.medication2.code = '040762'
medications.medication2.dateOfPrescription.dd = '15'
medications.medication2.dateOfPrescription.mm = '06'
medications.medication2.dateOfPrescription.yyyy = '2012'
medications.medication2.dateOfExpiration.dd = '06'
medications.medication2.dateOfExpiration.mm = '07'
medications.medication2.dateOfExpiration.yyyy = '2012'
```

Naloga vaje je, da na osnovi XML razčlenjevanja (*ang.* parsing) dano XML besedilo pretvorite v zapis v obliki Matlabove strukture, nato tej strukturi dodate izbrano podstrukturo in končno tako dobljeno strukturo pretvorite nazaj v XML besedilo.

1. Naložite XML besedilo iz datoteke sampleEHR.xml, pri čemer si pomagate z Matlabovimi funkcijami `fopen()`, `fread()` in `fclose()`.

2. Napišite funkcijo za pretvorbo XML besedila v strukturo:

```
function oStruct = xml2struct(iXml, iStruct, iTag),
```

kjer vhodni argument `iXml` predstavlja XML besedilo, `iStruct` trenutno strukturo, `iTag` pa celice z imeni vseh XML značk do trenutne značke. Izhodni argument `oStruct` predstavlja novo strukturo.

Zasnova funkcije naj omogoča rekurzivno klicanje v primeru gnezdenih XML značk, pri čemer je začetni klic enak `EHR = xml2struct(XML, struct, [])`. Rekurzija je metoda programiranja, kjer je rešitev danega problema sestavljena iz rešitev posameznih podproblemov. V praksi to izgleda tako, da klic funkcije opravimo znotraj funkcije same, pri čemer vsakič podamo drugačne argumente in zagotovimo zaustavitveni kriterij.

Uporabite to funkcijo za pretvorbo XML besedila v strukturo.

3. V dobljeno strukturo dodajte podstrukturo `medications.medication2`, ki je podana v uvodu.

4. Napišite funkcijo za pretvorbo strukture v XML besedilo:

```
function oXml = struct2xml(iStruct, iXml, iOffset),
```

kjer vhodni argument `iStruct` predstavlja strukturo, `iXml` trenutno XML besedilo, `iOffset` pa zamik trenutne XML značke. Izhodni argument `oXML` predstavlja novo XML besedilo.

Zasnova funkcije naj omogoča rekurzivno klicanje v primeru gnezdenih XML značk, pri čemer je začetni klic funkcije enak `XML = struct2xml(EHR, '', '')`. Za prehod v novo vrstico XML besedila uporabite ASCII kodo 10 oz. `char(10)`.

Uporabite to funkcijo za pretvorbo strukture v XML besedilo.

5. Shranite XML besedilo v datoteko updatedEHR.xml, pri čemer si pomagate z Matlabovimi funkcijami `fopen()`, `fwrite()` in `fclose()`.

Vprašanja

Poročilo vaje vsebuje lastnoročno (brez uporabe računalnika) zapisane odgovore na naslednja vprašanja, medtem ko so zahtevane slike natisnjene ter priložene poročilu.

1. Naštejte prednosti in slabosti zapisa podatkov v XML obliki ter vsako izmed njih na kratko obrazložite.

2. Napišite rekurzivno različico funkcije za izračun fakultete (faktoriele) naravnega števila N:

   ```
   function oValue = fact(iValue),
   ```

   kjer vhodni argument `iValue` predstavlja naravno število N, izhodni argument `oValue` pa fakulteto (faktorielo) števila N, torej $N!$, ki je določena kot:

   $$N! \quad = \quad N \cdot (N-1) \cdot (N-2) \cdot \ldots \cdot 2 \cdot 1 \quad = \quad \prod_{i=1}^{N} i \,.$$

3. Napišite iterativno različico funkcije iz vprašanje št. 2.

Dodatek

Odgovore na sledeče probleme ni potrebno prilagati poročilu. Služijo naj v razmislek ter kot vzpodbuda za boljše razumevanje vsebine.

Posplošite funkciji `xml2struct()` in `struct2xml()` tako, da bosta omogočali branje atributa `type`, ki predstavlja vrsto vrednosti, npr.:

- značka `<id type="number">123456</id>` naj predstavlja število `EHR.id = 123456`,
- značka `<dd type="string">12</dd>` naj predstavlja besedilo `EHR.dd = '12'`.

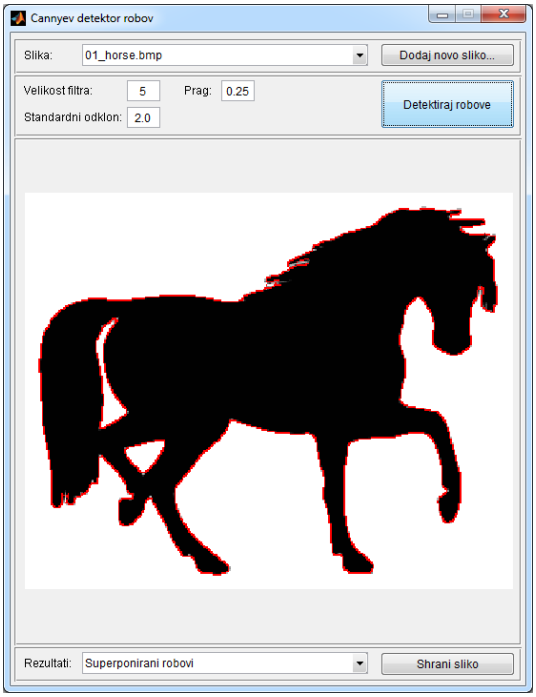
V primeru, da atribut ne obstaja, potem značka vsebuje gnezdeno XML besedilo. Za preizkus pravilnosti uporabite XML besedilo iz datoteke sampleEHR_attr.xml.

Vaja 7: Zaznavanje robov na slikah

Navodila

Zaznavanje robov na slikah je eden izmed najbolj pogostih konceptov na področju obdelave in analize slik, Cannyev detektor robov je eden izmed najbolj uporabljanih postopkov.

Pri tej vaji boste implementirali nekoliko poenostavljen Cannyev detektor robov, in sicer na podlagi danega uporabniškega vmesnika, ki omogoča nalaganje poljubne sivinske slike, izbiro parametrov detekcije robov (velikost in standardni odklon Gaussovega gradientnega filtra ter vrednost za upragovljanje), ter shranjevanje dobljenih rezultatov.



1. Dana je datoteka getGradientFilter.m z deklaracijo funkcije za generiranje gradientnega operatorja Gaussove funkcije:

```
function [oFilterX, oFilterY] = getGradientFilter(iN, iSigma),
```

kjer vhodni argument `iN` predstavlja velikost $N = 2M + 1$, `iSigma` pa standardni odklon σ gradientnega operatorja Gaussove funkcije velikosti $N \times N$, ki ga določa enačba:

$$\nabla N(x,y) = \nabla e^{-\frac{x^2+y^2}{2\sigma^2}} = \begin{bmatrix} -\frac{x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \\ -\frac{y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \end{bmatrix} = \begin{bmatrix} -x \\ -y \end{bmatrix} \frac{1}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} .$$

Izhodna argumenta `oFilterX` in `oFilterY` predstavljata gradientna operatorja $C_x(i,j)$ in $C_y(i,j)$ Gaussove funkcije v x in y smeri.

Slepo implementacijo dane funkcije nadomestite z delujočo implementacijo tako, da bodo izhodni argumenti dejansko predstavljali gradientni operator Gaussove funkcije glede na vhodne argumente, ki jih nastavljamo v uporabniškem vmesniku.

2. Dana je datoteka imageFiltering.m z deklaracijo naslednje funkcije za filtriranje slike:

```
function [oMapX, oMapY] = imageFiltering(iImage, iFilterX, iFilterY),
```

kjer vhodni argument `iImage` predstavlja vhodno sliko, `iFilterX` in `iFilterY` pa operatorja $C_x(i,j)$ in $C_y(i,j)$ za filtriranje vhodne slike v x in y smeri. Izhodna argumenta `oMapX`

in `oMapY` predstavljata filtrirano vhodno sliko $g_x(x, y)$ in $g_y(x, y)$ v smeri x in y. Za vsak slikovni element (x, y) opravite filtitranje kot:

$$g(x, y) = \sum_{i=-M}^{M} \sum_{j=-M}^{M} c(i, j) f(x + i, y + j) \,.$$

Slepo implementacijo dane funkcije nadomestite z delujočo implementacijo tako, da bodo izhodni argumenti dejansko predstavljali filtrirano vhodno sliko glede na vhodne argumente, ki jih pridobimo s pomočjo funkcije `getGradientFilter()`.

3. Dana je datoteka nonMaximaSuppression.m z deklaracijo naslednje funkcije za odstranjevanje nemaksimalnih vrednosti:

```
function [oMapX, oMapY] = nonMaximaSuppression(iMapX, iMapY),
```

kjer vhodna argumenta `iMapX` in `iMapY` predstavljata gradient $g_x(x, y)$ in $g_y(x, y)$ slike v x in y smeri, izhodna argumenta `oMapX` in `oMapY` pa gradient $g'_x(x, y)$ in $g'_y(x, y)$ slike v x in y smeri z odstranjenimi nemaksimalnimi vrednostmi. Odtstranjevanje nemaksimalnih vrednosti opravite tako, da slikovni element odstranite, če je amplituda gradienta v vsaj enem sosednjem slikovnem elementu, gledano v najbližnji diskretni smeri gradienta, večja kot v opazovanem slikovnem elementu.

Slepo implementacijo dane funkcije nadomestite z delujočo implementacijo tako, da bodo izhodni argumenti dejansko predstavljali gradient slike z odstranjenimi nemaksimalnimi vrednostmi glede na vhodne argumente, ki jih pridobimo s pomočjo funkcije `imageFiltering()`.

4. Dana je datoteka imageThresholding.m z deklaracijo naslednje funkcije za upragovljanje slike:

```
function oMap = imageThresholding(iMapX, iMapY, iThreshold),
```

kjer vhodna argumenta `iMapX` in `iMapY` predstavljata gradient $g'_x(x, y)$ in $g'_y(x, y)$ slike v x in y smeri z odstranjenimi nemaksimalnimi vrednostmi, `iThreshold` pa prag T za upragovljanje na intervalu med 0 in 1. Izhodni argument `oMap` predstavlja upragovljene vrednosti $u(x, y)$. Upragovljanje opravite tako, da slikovnim elementom z vrednostjo nad pragom priredite vrednost 1, ostalim slikovnim elementom pa vrednost 0:

$$u(x, y) = \begin{cases} 1; & \text{če je } f(x, y) > T \,, \\ 0; & \text{če je } f(x, y) \leq T \,. \end{cases}$$

Slepo implementacijo dane funkcije nadomestite z delujočo implementacijo tako, da bo izhodni argument dejansko predstavljal upragovljene vrednosti glede na vhodne argumente, ki jih pridobimo s pomočjo funkcije `nonMaximaSuppression()`, prag pa nastavite v uporabniškem vmesniku.

Preizkusite delovanje implementiranega postopka za zaznavanje robov na slikah 01_horse.bmp, 02_brain.bmp in 03_vertebra.bmp. Rezultate lahko shranite kot JPEG slike z ustrezno izbiro ter klikom na gumb "Shrani sliko" v uporabniškem vmesniku.

Vprašanja

Poročilo vaje vsebuje lastnoročno (brez uporabe računalnika) zapisane odgovore na naslednja vprašanja, medtem ko so zahtevane slike natisnjene ter priložene poročilu.

1. Za sliko 02_brain.bmp priložite originalno sliko, sliki gradientnega operatorja v x in y smeri, sliki gradienta v x in y smeri, sliko amplitude gradienta, sliko amplitude gradienta z odstranjenimi nemaksimalnimi vrednostmi, sliko upragovljenih vrednosti ter sliko s superponiranimi robovi. Zapišite izbrane vrednosti za velikost ter standardni odklon gradientnega operatorja Gaussove funkcije ter izbrano vrednost za prag pri upragovljanju.

2. Za sliko 03_vertebra.bmp priložite originalno sliko, sliki gradientnega operatorja v x in y smeri, sliki gradienta v x in y smeri, sliko amplitude gradienta, sliko amplitude gradienta z odstranjenimi nemaksimalnimi vrednostmi, sliko upragovljenih vrednosti ter sliko s superponiranimi robovi. Zapišite izbrane vrednosti za velikost ter standardni odklon gradientnega operatorja Gaussove funkcije ter izbrano vrednost za prag pri upragovljanju.

3. Obrazložite vašo izbiro vrednosti za velikost ter standardni odklon gradientnega operatorja Gaussove funkcije ter vrednosti za prag pri upragovljanju.

4. Katere poenostavitve smo upoštevali glede na "pravi" Cannyev detektor robov?

Dodatek

Odgovore na sledeče probleme ni potrebno prilagati poročilu. Služijo naj v razmislek ter kot vzpodbuda za boljše razumevanje vsebine.

Funkcijo `imageThresholding()` spremenite tako, da predstavlja vhodni argument `iThreshold` zgornji prag, spodnji prag pa določite kot polovično vrednost zgornjega praga in opravite histerezni upragovljanje. Histerezno upragovljanje ohranja vse slikovne elemente s sivinsko vrednostjo nad zgornjim pragom, slikovne elemente s sivinsko vrednostjo med spodnjim in zgornjim in spodnjim pragom pa v le v primeru, kadar imajo sosednji slikovni elementi sivinsko vrednost nad zgornjim pragom:

```
>>  % histerezno upragovljanje slike
>>  function oMap = imageThresholding(iMapX, iMapY, iThreshold)
>>      % zgornji in spodnji prag
>>      iHighT = iThreshold;
>>      iLowT = iHighT/2;
>>      ...
```

Vaja 8: Ujemanje zaporedij

Navodila

Pri tej vaji boste implementirali Needleman-Wunsch-ev algoritem za določanje globalnega optimalnega ujemanja zaporedij (npr. nukleotidov v DNK ali aminokislin v beljakovinah). Algoritem je poseben primer metode dinamičnega programiranja, s pomočjo katere lahko rešimo relativno kompleksne probleme tako, da jih razdelimo na lažje podprobleme.

1. Napišite funkcijo za določanje matrike ocen M in poti T za ujemanje zaporedij a in b:

   ```
   function [oScrM, oTrcM] = computeMatrices(iSeqA, iSeqB, iSubS, iSubM, iGapP),
   ```

   kjer vhodna argumenta `iSeqA` in `iSeqB` predstavljata zaporedji a in b, `iSubS` predstavlja zastopane znake v izbranem vrstnem redu (npr. `'AGCT'`), `iSubM` predstavlja substitucijsko matriko S z znaki v enakem vrstem redu, `iGapP` pa predstavlja kazen za vrinjeno mesto P. Izhodna argumenta `oScrM` in `oTrcM` predstavljata matriko ocen M in matriko poti T.

   Matriko ocen M ustrezno inicializirajte z začetnimi vrednostmi (ničle), matriko poti T pa z začetnimi smermi. Oceno $M(i,j)$ in smer $T(i,j)$ določite s pomočjo rekurzivne formule:

   $$M(i,j) = max \begin{cases} M(i-1,j-1) & + & S(a(i),b(j)) & \nwarrow & \text{(smer diagonala oz. D)} \\ M(i,j-1) & + & P & \leftarrow & \text{(smer levo oz. L)} \\ M(i-1,j) & + & P & \uparrow & \text{(smer gor oz. U)} \end{cases}$$

   V primeru, da je več vrednosti hkrati največjih, v matriko poti T vpišete samo eno smer, pri čemer se držite predpisanega vrstnega reda, in sicer najprej smer diagonala ($\nwarrow$ oz. D), nato smer levo ($\leftarrow$ oz. L) in nazadnje smer gor ($\uparrow$ oz. U).

2. Napišite funkcijo za določanje optimalno ujemajočih se zaporedij:

   ```
   function [oSeqA, oSeqB] = computeSequences(iSeqA, iSeqB, iTrcM),
   ```

   kjer vhodna argumenta `iSeqA` in `iSeqB` predstavljata zaporedji a in b, `iTrcM` pa matriko poti T. Izhodna argumenta `oSeqA` in `oSeqB` predstavljata optimalno ujemajoči se zaporedji a in b.

   Optimalno ujemajoči se zaporedji poiščete s pomočjo sledenja glede na smeri v matriki poti T, pri čemer s sledenjem pričnete v zadnjem elementu matrike.

3. Napišite funkcijo za izračun ocene optimalnega ujemanja:

   ```
   function oScr = computeScore(iSeqA, iSeqB, iSubS, iSubM, iGapP),
   ```

   kjer vhodna argumenta `iSeqA` in `iSeqB` predstavljata optimalno ujemajoči se zaporedji a in b, `iSubS` predstavlja zastopane znake v izbranem vrstnem redu (npr. `'AGCT'`), `iSubM` predstavlja substitucijsko matriko S z znaki v enakem vrstem redu, `iGapP` pa predstavlja kazen za vrinjeno mesto P. Izhodni argument `oScr` predstavlja oceno optimalnega ujemanja.

Preverite implementacijo algoritma tako, da določite optimalno ujemanje nukleotidnih (DNK) zaporedij $a =$ GGATCGA in $b =$ GAATTCAGTTA, za katere je bila rešitev podana pri pripravah na laboratorijske vaje. Uporabite substitucijsko matriko S in kazen za vrinjeno mesto P:

S	A	G	C	T
A	2	−1	−1	−1
G	−1	2	−1	−1
C	−1	−1	2	−1
T	−1	−1	−1	2

$P = -2$

S^*	A	G	C	T
A	2	1	−1	−1
G	1	2	−1	−1
C	−1	−1	2	1
T	−1	−1	1	2

$P^* = 0$

Vprašanja

Poročilo vaje vsebuje lastnoročno (brez uporabe računalnika) zapisane odgovore na naslednja vprašanja, medtem ko so zahtevane slike natisnjene ter priložene poročilu.

1. Določite optimalno ujemanje nukleotidnih (DNK) zaporedij $a =$ ACA in $b =$ CGACT, pri čemer uporabite substitucijsko matriko S in kazen za vrinjeno mesto P. Zapišite tudi matriko ocen M, matriko poti T (s puščicami in z označeno optimalno potjo) ter oceno optimalnega ujemanja.

2. Določite optimalno ujemanje nukleotidnih (DNK) zaporedij $a =$ CTCTAGCATTAG in $b =$ GTGCACCCA, pri čemer uporabite substitucijsko matriko S in kazen za vrinjeno mesto P. Zapišite tudi oceno optimalnega ujemanja.

3. Določite optimalno ujemanje zaporedij iz vprašanja št. 1, pri čemer uporabite substitucijsko matriko S^* in kazen za vrinjeno mesto P^*. Zapišite tudi matriko ocen M, matriko poti T (s puščicami in z označeno optimalno potjo) ter oceno optimalnega ujemanja.

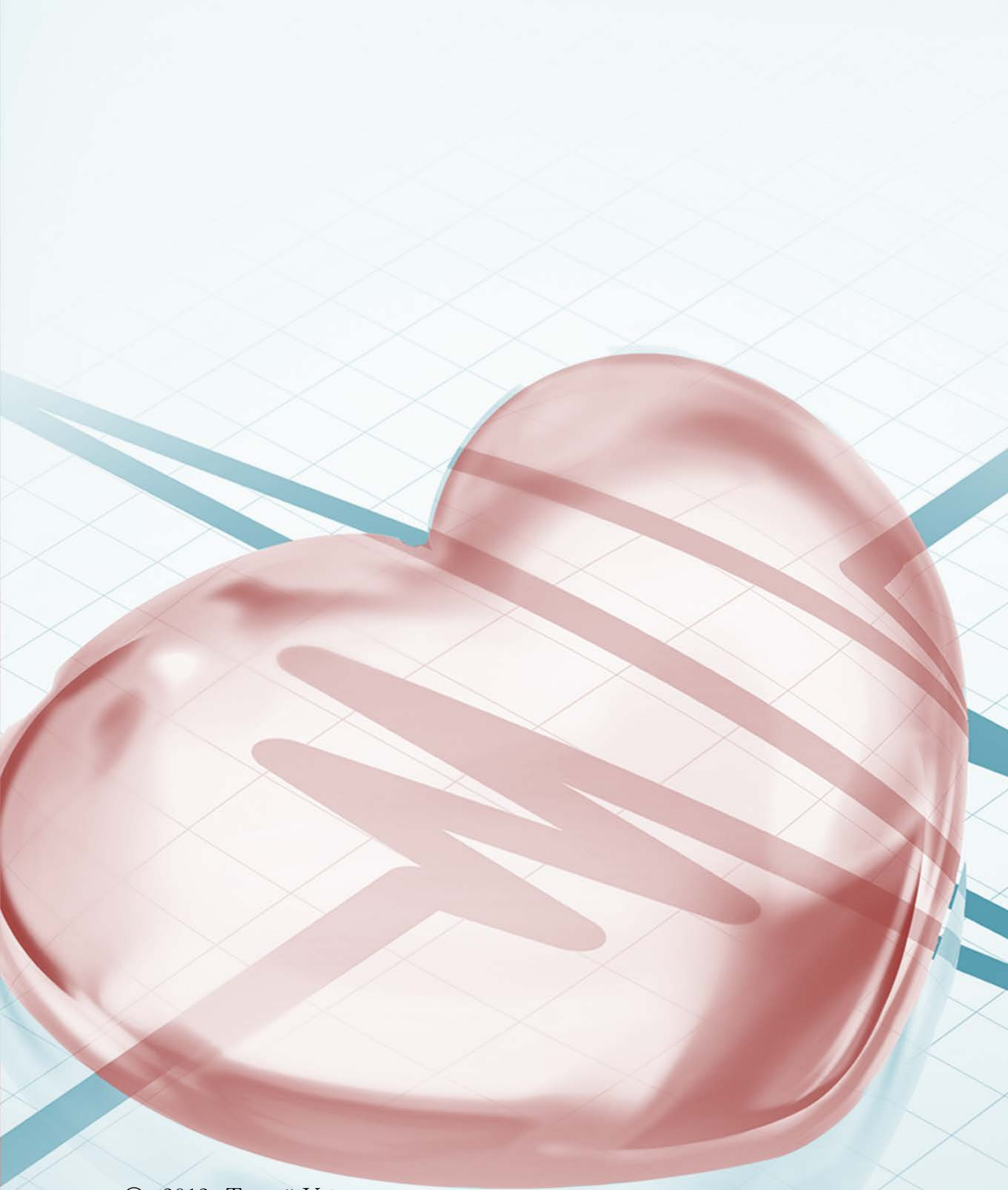
4. Določite optimalno ujemanje zaporedij iz vprašanja št. 2, pri čemer uporabite substitucijsko matriko S^* in kazen za vrinjeno mesto P^*. Zapišite tudi oceno optimalnega ujemanja.

5. Katere vidike smo zanemarili pri obstoječi implementaciji?

Dodatek

Odgovore na sledeče probleme ni potrebno prilagati poročilu. Služijo naj v razmislek ter kot vzpodbuda za boljše razumevanje vsebine.

Pri obstoječi implementaciji algoritma smo zanemarili dejstvo, da lahko obstaja več optimalnih ujemanj danih zaporedij, ki bi izhajala iz več možnih izbir v matriki poti T zaradi hkratnih največjih vrednosti v rekurzivni formuli za iskanje vrednosti matrike ocen M. Popravite obstoječo implementacijo tako, da bo omogočala določanje vseh obstoječih optimalnih ujemanj zaporedij.



http://lit.fe.uni-lj.si/BMI/
http://lit.fe.uni-lj.si/gradivo/BMI-LabVaje-slo.pdf