# Tätigkeitsbeschreibungen Algorithmen

#### Fehlerprotokolle, Fragen, Dokumentation

* mit airfoil\_self\_noise Testset, Startwert [C,eps] = [1,0]
* quadprog dauert sehr lange
* erreicht immer die maximalen 500 Iterationen
* BB-Algorithmus bricht bereits nach einem Schritt ab
* mit airfoil-set gekürzt auf 100 Datensätze und 3 folds (67 je Fold)
* quadprog löst in wenigen Sekunden
* erreicht nicht die 500 Iterationen

### Probleme:

**1 Problem:** wenn constraints C>0, eps>=0, dann sinnlose Ergebnisse (xi,d zu klein...)

Wenn diese Constraints nicht da, dann C negativ, dadurch Supproblem nicht konvex

--> dann hängt alles in Endlosschleife

auch wenn nicht bound constraints, sondern in inequality constraints drin (sogar schlimmer)

**LÖSUNG:** bound constraints waren falsh einprogrammiert, ist jetzt gelöst

**2 Problem:** Lower level problem braucht > 500 iterationen

**LÖSUNG:** "Constraints" zu eng gewählt -> ungenauere Lösung zulassen, dann wird auch gelöst

macht auch Sinn, da inexact Algorithm genau damit umgehen können sollte -> dafür ausgelegt

(Unterschied zu Bundle Subproblem, welches laut Theorie sehr exakt gelöst werden muss)

**3 Hauptproblem:**

- Stopping Conditions sehr unterschiedlich (Verhalten des Algos)

Unterprobleme:

1. Algorithmus stoppt entweder sehr schnell oder gar nicht

Lösungsansätze:

- Welche Stopping Conditions bringen gleiches Ergebnis?

1, 4, 5: erfolgreicher Stopp nach 11 Iterationen (8 null steps);

x\_hat = (42.2113;21.1455); auch hier eta recht groß

2, 3, 6: x\_hat = (42.5234;56.8496); k = 310; i\_null = 294;

dann c erstes Mal kleiner 0 weil b in einer Komponente falsch

- Idee: Stopping Conditions 1 vs. 2 zeigen: 1/t\*d ~= G -> klar, da Constraints im Subproblem

1/t\*d = G+v

-> Formulierungen der Stopping Conditions prüfen:

äquivalent (auch numerisch):

**LÖSUNG:** Stopping conditions waren nicht richtig implementiert G statt G+v (Unterschíed durch

constraints im Subproblem --> Reformulierung, Ergebnisse siehe Unterfrage 3)

1. C bzw. c (und delta) irgendwann < 0

Lösungsansätze:

* + prüfe, wann das auftritt: für air-datensatz (kurz) nach 310 Iterationen
  + warum? b stimmt nicht

! auffällig: t ist da bei t\_min angekommen -> bessere t-Veränderung???

! auffällig: in x enthält lauter gleich Werte -> prüfen, wie das sein kann

! auffällig: eta wird sehr groß; aber kein direkter Zusammenhang zwischen großem eta und negativem c feststellbar; eta springt auch viel zw. 2 und 1e~29

* alpha nicht "nah genung" bei 0 für rausnehmen aus bundle?

1. Test der neuen Stopping Conditions

* 3) stoppt genau nach 309 iterationen, 292 null steps; ZUFALL??? mit x\_hat = (42.5234;56.8496) (wie vorher), eta auch groß; anscheinend alles andere wie vorher
* 3a) alles genau wie oben
* 6), 6a) Stoppen jetzt wie 1)

**LÖSUNG:** einzig anderer Stopping test ist der von conv, inex 🡪 Unterschied verstehen?

* Vergleiche echtes Lernergebnisse die die jeweils gefundenen x\_hat = [C,eps] ergeben

**4 Hauptproblem:**

Quadprog löst Regression Problem seeeeeeeehr langsam (20 min) außer für Werte von eps = 56.8496 +-2; dann Lösung in <4sek

Lösungsansätze:

Unterprobleme:

1. Datensatz überhaupt nicht linear trennbar???

* Problem noch mal mit anderem Algorithmus lösen:
  + gelöst für erste 99 Datensätze des airfoil-sets, [C,eps] = [42.5234/3; 56.8496]  
    Ergebnis mit quadprog: W = [4.595762226113112e-05;-0.006082862314230;0.023484732332632;2.281886237215502;2.550177449925015e-04]  
    Ergebnis mit qpas: W = [4.595762226113345e-05;-0.006082862314230;0.023484732332632;2.281886237215502;2.550177449925003e-04]  
    Unterschied: 10^-15 🡪 eigentlich kein Unterschied
  + gelöst für erste 99 Datensätze des airfoil-sets, [C,eps] = [42.5234; 21.1455]  
    Ergebnis mit quadprog (unsicher): W = [0.001334505894522;2.200361584083511;1.279705679689002e+02;1.673168266205617;-2.931031012568054e+02]  
    Ergebnis mit qpas: Löst in Sekundenbruchteilen!  
    W = [-0.001586099220891;-1.817616651687124;2.098842793445674e+02;1.226857661128482;2.638118691075275]  
    Unterschied: 10^2, Unterschied W1\_qpas zu W2\_qpas ebenfalls 10^2

„Lösung 1“: Versuche erst mal mit Classification Datensätzen weiterzukommen

Wahrscheinlich keine lineare Regression möglich für airfoil Datensatz 🡪 siehe: <http://kernelsvm.tripod.com/>

„Lösung 2“: Schauen, ob schnell ein linearer (Regressions-)Datensatz gefunden werden kann

* Datensätze gefunden, die laut Quelle für lineare Regression geeignet
* Bei einem Test brach Algorithmus schon nach erstem Schritt ab – Warum???

„Lösung 3“: Datensatz vorher war nur nicht standardisiert???

**Klassifizierung anschauen**

* Datensätze aus Kunapuli benutzen (für Vergleichbarkeit)
* Schauen, ob Classification auch über least sqares Lossfunktion ausgerechnet werden kann 🡪 Vergleich mit Kunapuli
  + Inner Level Zielfunktion bei Kunapuli:  
    Unterschiede zu Moore:  
    λ als Regularisierer anstatt C 🡪 numerisch stabiler [Kunapuli, S.47]  
    ξ-Reformulierung hat zusätzlich Nebenbedingung ξ ≥ 0 🡪 bei Moore nicht, weil ξ in Zielfunktion quadratisch vorkommt?
  + Outer Level Zielfunktionen bei Kunapuli:  
    Durchschnitt der falsch klassifizierten Punkte: 🡪 diskontinuierliche Funktion; kann als Optimierungsproblem umgeschrieben werden 🡪 KKT-Bedingungen als Nebenbedingungen der Zielfunktion 🡪 ABLEITUNG???  
      
    Auch möglich: Hinge Loss
  + Logistic loss possible: 🡪 log{1+e^[-y(x’w-b)]}
  + Mean squared error possible? 🡪 [1-y(x’w-b)]^2 🡪 passt nicht gut zu anderen Loss Functions; eher nicht nehmen 🡪 laut Internet möglich, aber einige “Probleme”
* Löse für Pima-Set mit Moore inner Level, hinge-loss outer Level; vergleiche wie nah an Ergebnissen von Kunapuli

**Hauptproblem:** Bundle Algorithmus bricht schon nach einem Schritt ab

* Ursache: lambda (=d) immer an unterer Grenze der Bound Constraints 🡪 denn, unconstrained ist lambda ~-2e-8 🡺 wenn constrained auf 0: lambda = xi = 0 🡪 Abbruch
* Wenn unconstrained: lambda und xi negativ 🡪 delta im ersten Schritt positiv 🡪 läuft auch weiter, aber Theorie eben falsch…
* Wenn constrained auf feste Zahl > 0: lambda diese Zahl, xi positiv 🡪 delta negativ (1. Schritt)
* Wenn Subproblem mit quadprog gelöst wird 🡪 auch bei constrained auf 0 is d minimal positiv 🡪 xi auch 🡪 erster Schritt delta negativ

Mögliche Fehler:

* Fehler in einer der Matrizen des Subalgorithmus 🡪 unwahrscheinlich, da diese immer gleich (bis auf constraints) und in anderen Algorithmen OK
* !!! Mit rand laufen lassen, sodass immer unterschiedliche Werte!!! 🡪 Vergleichbarkeit
* Neues delta 🡪 Algorithmus läuft sehr schnell, aber erreicht 1000 Iterationen!
* Macht anscheinend keine!!!!!!!!! Null steps!!! 🡪 aber wenn auf pima set gestet 🡪 wieder sofort nach einem Schritt Abbruch

Erkenntnisse:

* 1. Schritt: Wenn Minimum bei negativem x 🡪 d ~ 0 (d=lower bound) 🡪 klar, richtig 🡪 falshc, siehe nachfolgenden Punkt
* !!!!!!!!! nicht d sonden lambda = x\_hat(k+1) > 0 🡪 d > -x\_hat
* Obiges und deltas ausgebessert 🡪 läuft jetzt, allerdings immer k=4, i\_null=3 🡪 dann Abbruch mit Startwerten 🡪 f ist bei startwerten 1,10,100 auf 10^-15 gleich
* Schauen, ob sich was ändert, wenn statt 0,1 🡪 +-1 als Classifier
  + K, i\_null ändert sich nicht; x\_hat ändert sich nicht; Gefühl, dass Unterschiede in f-Werten etwas größer
  + 🡪 ändere Datensätze auf +-1 Classifier
* Möglichkeiten: mit pima set und diesen Lambda werten lernen und fehler vergleichen
  + Datensätze vorbereiten für Tests 🡪 festes hold out set, feste training partitions 🡪 so wie in Kunapuli
  + Datensätze, Permutation… 🡪 speichern
  + Mit lambda = 1/3\* 1,10,100 W’s ausgerechnet (über hingelossquad, qpas 🡪 so, wie lambdas in bilevel-Algo berechnet) 🡪 Unterschiede in den W’s
  + Schon funktionswerte in Optimierung viel zu klein ~1 statt ~23; ist beim Testen dann genauso 🡪 irgendwas grundsätzliches falsch
  + Versuche Funktion zu programmieren, die falsche Klassifizierungen zählt (und auf Datensatzgröße normiert)
  + Alle Datensätze von Kunapuli mal durch Matlab jagen und schauen, ob Test-Error von Kunapuli und Prozentsatz in Matlab zusammen passen
    - Pima: Test error Kun: ~24; Trefferquote Matlab: 77.6%
    - Cancer: Test error Kun:4; Trefferquote Matlab: 96.6%
    - Heart: Test error Kun: ~16; Trefferquote Matlab: 82.2%
    - Ionosphere: Test error Kun: ~23; Trefferquote Matlab: 87.7%
    - Scheint einen Zusammenhang zu geben 🡪 Testerror scheint in etwa die Anzahl der falsch klassifizierten pro hundert zu sein
  + Trainiere nur ein Mal (Postprocessing Funktion) mit händisch eingegebenem Parameter lambda auf Trainingsset🡪 berechne Fehler auf Validationset  
    🡪 wenn Fehler >= 🡪 gut  
    🡪 wenn Fehler < Error function oder Trainingsproblem falsch
    - **Pima:** Mit verschiedenen lambda (0.1, 1, 10, 100) trainiert: W’s alle ähnlich aber nicht gleich; Norm wird mit steigendem Lambda kleiner 🡪 passt zu Zielfunktion  
      Fehler: 1.1… alle ähnlich, viel zu klein; sinken mit steigendem lambda
    - **Heart:** W’s größere Unterschiede als bei Pima, aber man kann sich Verläufe erklären  
      Gleiches Verhalten beim Fehler wie oben (1.4-1.7)
  + Funktion geschrieben, die missklassifizierte Objekte zählt  
    W’s so wie oben errechnet 🡪 Fehlerraten:
    - **Pima:** 51-59% besser mit steigendem lambda 🡪 bei beliebigen Werten und klassifizierung in 2 Klassen eine Fehlerquote von 50% zu haben ist genau richtig, denn Wahrscheinlichkeit, dass zufällig richtig/falsch = 50%
    - **Heart:** 76-80%
  + Hinge loss (Berechnung für w, b) neu programmieren (Kunapuli-Funktion nehmen??)
    - Fehler in den constraints gefunden 🡪 Vergleich: neue constraints ergeben w, b dann ergeben alte –w, b 🡪 Missklassifizierungen natürlich viel höher
    - Hinge loss und Missklassifizierungsfehler passen immer noch nicht so recht zusammen
    - Interessant: bei großer Veränderung in lambda nur wenig Veränderung im Fehler…
    - **Pima**: Fehler besser als in PhD Thesis 🡪 bedenklich!
    - Hinge wie in Kunapuli: ergibt andere w  
      Constraints auf 10^-31 erfüllt; bei quad genau erfüllt  
      Fehler fast immer geringer als bei quad; ähnlich zu quad-Fehlern  
      w seht anders wenn mit quad ; größerer Unterschied, wenn man hinge anstatt misclassification loss anschaut
    - Hingeloss upper bound on misclassification 🡪 muss immer größer sein als misclassification loss  
      Hinge loss ist richtig  
      passt zumindest zu CV-error in Kunapuli 🡪 dort hinge auch immer viel Größer  
      HIngequad in Kombination mit Misclassification error auch vollkommen in Ordnung; mit hinge err sind die Fehler doch recht groß
    - Achtung: In Kunapuli-Thesis eine andere Funktion zur Berechnung des Test errors genommen 🡪 diese programmieren und vergleichen  
      🡪 ist gleiche Funktion wie misclassification 🡪 kommen auch dieselben Fehlerzahlen raus
    - Großen Test geschrieben 🡪 Fehler immer noch viel zu klein  
      evtle Gründe:  
      Summe Falsch  
      Fehler wegen Feature selection so groß (kann eig nicht sein, da Matlab ähnliche Ergebnisse) und unc. Grid ja auch so
  + Getestet mit Skript aus LTH Hausaufgabe
    - W, b Bestimmung richtig (nur: sie nehmen –b, ich b)
    - Fehler wenn man –b einsetzt falsch  
      Wenn b auch falsch, sieh aber besser aus  
      Hinge-Fehler zu groß (kann aber sein, dass das richtig), misclass Fehler zu klein  
      y-labels sind 1,0 🡪 passt nicht zu meiner Loss-function?  
      geändert: passt
    - B angepasst, allg so angepasst dass identisch mit Situation in HA  
      🡪 Wb-Berechnung stimmt sowohl für hinge als auch hingequad  
      🡪 Misclassification Error immer noch falsch 🡪 in meisten Testfällen zwar gleicher Fehler, aber es werden nicht die gleichen Punkte als missklassifiziert eingestuft 🡪 Vertippt
    - Missklassifikationsfehler stimmt
    - Hinge loss stimmt
    - Robustheit gegenüber outliern so wie in ex\_3\_2\_4 gegeben
  + LTH Classifier (Fisher und log regression) auf Kunapuli Testdaten anwenden
    - Logistic regression konvergiert nicht  
      Fisher ergibt deutlich schlechtere Fehler und kann für ionosphere nicht lösen
    - Bei einfachem Durchlauf stimmen jetzt meine Fehler (sind ungefähr gleich, mal schlechter mal etwas besser)
    - 20 Testläufe: Fehler passen eigentlich ganz gut, bis auf ionosphere –Fehler 🡪 viel zu gering
    - Ionosphere in Matlab-App testen: 86,9% Trefferquote 🡪 passt zu meinen 13,9% Fehlerquote  
      Wenn erste Spalte nicht Normalisiert sogar 88,3% Trefferqote (MATLAB-App); bei mir Fehler etwas höher: 14-15%
  + Matlab svm classifier auf Kunapuli Data
    - Bei pima w recht ähnlich, bei ionosphere gar nicht 🡪 allg schlechte Vergleichbarkeit
  + Obige Ergebnisse alle mit lambda = 10
  + Lambda = 100 (20 Durchgänge, LTH-Skript): Änderung in Fehlern (0,1-3%)  
    Bei lambda = 10000 bis 40%
  + test von bilevel bundle anhand der LTH-Beispiele
    - Vergleich der Zwischenergebnisse im Algorithmus mit den entsprechenden post\_pro-Funktionen (diese in LTH-Skript getestet)  
      Wb stimmt, f stimmt  
      1. Schritt: Unterschied zwischen x\_hat, x\_hat+d so gering, dass Ableitungen, Funktionswerte… quasi gleich 🡪 geht dann so weiter
    - Testen, ob bei pima und hinge-loss Unterschied zwischen f’s bei lambda = 10 und 100 wirklich so gering   
      JA, bei pima-Datensatz Fehler nur einen Unterschied von 10^-15 – 10^-13 für Verschiedene lambdas 🡪 habe (wie Algorithmus auch???) den Fehler auf dem trainings-set berechnet 🡪 daher der kleine Unterschied 🡪 JA
    - was passiert, wenn man auf validation set den Fehler berechnet?  
      Einzeltest ergibt: verschiedene f für verschiedene lambda (0.3 Unterschied)
    - Wie macht es der Algo? 🡪 Algo muss mit mind. 2 folds Arbeiten, sonst macht es keinen Sinn 🡪 einfach nur mehr folds ergibt noch keie Änderung 🡪 wie werden fehler… Berechnet  
      🡪 kann der Gradient überprüft werden
  + Test mit 3 Folds:
    - W’s im ersten Schritt für verschiedene Folds recht unterschiedlich  
      b’s ähnlich aber unterschiedlich
    - Abstand zwischen Funktionswerten schon nicht mehr so klein, aber immer noch 10^-5
  + Problem: weiß nicht, welches lambda korrekt 🡪 ebenfalls grid search implementieren???

**Warum Cross Validation?**

* Methode gegen Overfitting 🡪 nicht nur „in-smaple-error“, sondern auch Generalisierungsfehler wird abgeschätzt
* Ist auch? „tragende Idee“ des Bilevel Problems 🡪 der Fehler sollte nicht auf dem Trainings-set berechnet werden 🡪 mind. 2 Folds  
  in der Praxis und für eine sinnvolle Vergleichbarkeit der Fehler unerlässlich

#### Plot upper level objective (hinge loss) for different lambdas

Lambda: 200 points in logarithmic scale vrom 10^-4 to 10^4

Script in Matlab: plot\_hinge\_loss\_lambda

* Matlab script written for hinge loss and hingequad loss for wb calculation, average over 20 times
* Looks like Error monotonically increasing –Why???
  + Pima
    - Used post-processing error for comparison 🡪 same error
    - Min/max are first/last entry of the vector
    - Not the case for misclassification error 🡪 min for lambda(143) = 51.114334834401653 and function value 0.241761363636364  
      (without average: lambda = 7.316807143427193 with function value 0.227272727272727)  
      (max: lambda= 3.612342699709430e+03)
    - Difference in fp value for consecutive values lies between 10^-8 and 10^-3 (10^-3: lambda(163-164) difference here 32) 🡪 almost same values as for only one try (here lambda 161-162)
    - Pluged 0 in manually (1 try): gives for hinge loss actually the min 🡪 approximately same values as for small lambda values  
      if lambda = 0: norm(w) not considered 🡪 could be possible that for not linearly separable set w already fixed by “wrong lying” points and margin not needed
    - Objective function (hinge) monotonically increasing for the given lambdas
  + Cancer
    - Postpro function and ul\_obj same value
    - Min at lambda(110) = 2.409403560239527 with f(110) = 0.105337973566117  
      (only one try: Maximum is last value: minimum at lambda = 3.180625692794119 with function value 0.109559824920219)
    - For misclassification loss: lambda(133) = 20.255019392306664 and m(133) = 0.031376975169300  
      (one try lambda: 46.595256686646778 and function value 0.024830699774266)
    - Difference of successive values:  
      min: 10^-5  
      max: 0.017060050611011 lambda: 126,5…
    - Graph very narrow and steep valley between 0 and ~ 1000 with min between 2 and 3
  + Heart
    - Min at lambda(1)=10^-4, max at lambda(200)=10^4 🡪 monotonically inceasing  
      f(1) = 0.475958432003064
    - Misclassification: min lambda(156) = 170.277… m(156)= 0.152469135802469  
      max for lambda(200)
    - Difference in values: min: 1.389634007864515e-08; max: 0.011512878294623 at lambda(169) 🡪 diff: 55.014476823825362
    - Monotonically increasing
  + Ionosphere
    - Min at lambda(115) = 3.827494478516315 f(115)= 0.353617073076557  
      max at lambda(1)=10^-4; f= 1.266166258875075
    - Misclassification: min at lambda(113) = 3.180625692794119 with 0.121171171171171  
      max at lambda(196)= 6.905513520162331e+03 with 0.371621621621622
    - Difference in values: min: 4.009462612886150e-05  
      max: 0.016824416553723, lambda-diff(41): -3.933220952867574e-04
    - Steep, narrow valley like for cancer set; mit ~at lambda=4
  + Conclusions:
    - Upper level objective and post processing function for error calculation are the same
    - Change in objective function very small 🡪 hard for algorithm to optimize
    - Pima and heart and ionosphere and cancer have similar plots and behavior

#### Test gradients:

* Df/dlambda:
  + First try: not at all similar
  + Now that dw/dlambda right ok to 10^-4 (because of optimization routine???)
  + Df/dw shows: can only be incorrect because of dw/dlambda
* Df/dw:
  + Program to see if df only wrong because of dw…
  + Was ok for eps 10^-8 (up to 10^-6)
  + Tested that respective part of df/dlambda is right
* Dw/dlambda:
  + Forst try: not at all similar
  + Was wrong, now right
  + Works to 10^-3 🡪 because of optimization routine within calculation???
* Df and dw/dlambda
  + No change in the 10^-4/10^-3 no matter if eps 1 or 10^-15  
    for dw difference of 10^-1 if take 10^-0 or 10^-15
  + Think this is because of the optimization step
* Testing for different datasets to see if sensitive to them
  + Pima: see results above
  + Cancer: similar to pima
  + Heart: similar results to pima
  + Ionosphere: dw similar to pima; df only 10^-3 🡪 but consistently  
    sometimes also dw up to difference 1 (10^0)

Again testing because of “nonquad-Version”

* Tests do not work anymore (too high differences) (12.04.2017)
  + Df/dw solved 🡪 scaling was not in all functions  
    slightly other eps necessary because of scaling
  + Same derivatives: ok 🡪 !need to change some lines in Function because dwb=1
  + Dw/dlambda 🡪 was so bad because of data set?
  + Dfdlambda 🡪 still not so good 🡪 scaling!!!
* Tests “without quad” not working 🡪 check derivative and implementation by hand

#### Algorithm testing with cancer and ionosphere

* Ionosphere:
  + Eta about 15
  + Even if starting value very close to optimum: uses up all steps
  + 0 null steps 🡪 all steps are serious steps???
  + Tendency in the right direction if coming from above 🡪 allow bigger steps???  
    if starting point below optimum: wrong searching directions
  + Delta in last step at least near
* Cancer:
  + Very similar eta values (too similar?)
  + All steps, 0 null steps
  + Delta comes near stopping condition
  + From above right tendency; from below: right tendency but ***really*** little progress, eta quite high although function very convex in this area!
* Testing bigger t’s
  + Algorithm stops now
  + Differences between x\_0 and x\_opt:  
    t\*1,0001; cancer; x\_0=2; x\_kmax = 1.888005211435859  
    t\*1,01; cancer; x\_0=2; x\_675 = 1.328194392397111  
    t\*1,1; cancer; x\_0=2; x\_97 = 1.328194946523427  
    t\*1,5; cancer; x\_0=2; x\_33 = 1.317800247709140  
    t\*1,5; cancer; x\_0=10; x\_41 = 1.320089922885692  
    t\*1,5; cancer; x\_0=1000; x\_75 = 1.005323217632659  
    t\*1,5; cancer; x\_0=0.001; x\_34 = 0.788835362658012  
    t\*1,5; cancer; x\_0=0.1; x\_32 = 0.788833165856563  
    There seem to be local minima at the above seen x-points 🡪 f\_values:  
    0.104886432409386 🡪 1.3

0.104886088302081  
0.104886365700891  
0.104818800676197 🡪 1.005  
0.104825422465473  
0.104825422626028 🡪 0.7  
!!!!! this is only over one instance, not averaged!!!  
ionosphere: x = 0.543171787728030 from below and above 🡪 stepsizes do get big, but from theory no problem

* For pima result is x=0 🡪 matches graph
* Heart: x=0 🡪 matches graph
  + Results of the algorithm do not seem to match the lambdas shown in the plots
* Scaling the objective function:
  + With scaling \*1000 a lot more steps needed but still same results for the x-value 🡪 they seem to be right  
    cancer seems to stick with 1.005 most of the times but also 1.3 and 0.7 if special starting values

Interesting fact: computation time really much less than in PhD-thesis

#### Final script

* Errors: Computation time really low  
   0 values where there should not be  
   error for ionosphere too low
  + Computation time: comment all tic toc 🡪 better but still really fast
  + Errors: optimized last w,b on test set, not on trainings set 🡪 errors ok now
* Only seldom case that it doesn’t converge (seems only to be the case for ionosphere??? 🡪 test this
  + K=1000 also for cancer and heart

## Noll-Version von Hare-Bundle-Algorithmus

Conparison of Hare algorithm and Noll Version

For time comparison run each algorithm 2-3 times

Optimum for all algorithms at 0; stopping condition from conv-inex (with d, C); tol = 1e-6

Blue: Changed: Q+1/t\*I is always pos semi def; not manipulating t but Q on its own (with q)  
Green: Q+1/t\*I regulation by t  
Red: new stopping criterion 🡪 C<tol and (Q+1/t\*I)\*d<0  
Purple: new stopping condition (more theory behind it) delta = C+d’\*(Q+1/t\*I)d < tol  
Orange: bounded Q (to 10^5)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algo | Test function | X0 | F | X | time | iter | Null steps | T end  T averag | | Comments |
| Hare  qpas | F1 | Ones(5,1) | 2.0726e-07  2.0726e-07 | 1.0e-07 \*  0.4246  -0.2163  -0.3712  -0.2507  -0.1742 | 0.3234  0.0518 | 20  20 | 6  6 | 0.2805  0.1785  0.1785 | |  |
| Noll | F1 | Ones(5,1) | 0.0164  1.3898e-04  3.1815e-06  8.2632e-08  2.0966e-07 | 0.0015  -0.0003  0.0024  0.0005  0.0013 | 0.0672  0.4007  0.1347  0.0718  0.0431 | 25  79  48  52  35 | 13  49  16  8  12 | 0.0490  0.0258  0.0310 | | t < 10^-6 |
| Hare | F2 | Ones(5,1) | 2.1955e-08  8.0020e-14 | 1.0e-04 \*  0.2237  0.2165  0.2098  0.2382  0.2157 | 0.6777  0.1344 | 34  45 | 14  19 | 0.2617  0.1555  0.1555 | |  |
| Noll | F2 | Ones(5,1) | 60.8502  2.5117e-09  5.4736e-14  4.1410e-07  4.1410e-07 | -0.9072  -1.4085  -1.5796  -1.1271  -1.3364 | 0.0688  0.5330  0.1392  0.1125  0.1233 | 8  87  60  53  53 | 7  30  11  11  11 | 0.0360  0.0215  0.0215 | | Alpha totally wrong (=0)🡪 C wrong 🡪 stop  Alpha 10^-7  Often 10^-9  Wrong x!!!  Alpha 10^-9  X wrong as before  Alpha, X again |
| Hare | F3 | Ones(5,1) | 2.2822e-08  1.8460e-10 | 1.0e-08 \*  0.8200  0.4651  -0.3206  -0.1358  0.8121 | 0.5741  0.0847 | 33  33 | 9  9 | 0.8892  0.6554  0.6554 | |  |
| Noll | F3 | Ones(5,1) | 0.0011  3.3152e-06  1.0120e-07  5.5423e-07  7.2449e-07 | 1.0e-03 \*  -0.5459  0.3178  0.3470  0.2247  -0.3640 | 0.1418  0.1907  0.0730  0.0489  0.0634 | 19  59  36  48  40 | 5  31  15  6  19 | 0.4207  0.1268  0.1633 | | q >10^6 alpha 10^-10  t < 10^-8 |
| Hare | F4 | Ones(5,1) | 7.9240e-08  4.6523e-07 | 1.0e-07 \*  0.0728  -0.1077  0.1379  0.0425  -0.0720 | 0.4245  0.0780 | 25  27 | 7  10 | 0.4653  0.2413  0.1456 | |  |
| Noll | F4 | Ones(5,1) | 0.0443  0.0014  7.8891e-07  7.8891e-07  1.1218e-07 | 0.0070  0.0071  0.0036  -0.0015  -0.0016 | 0.1164  0.1182  0.1175  0.0449  0.0556 | 12  21  42  42  29 | 5  10  17  17  10 | 0.1174  0.0348  0.0464 | | Alpha = 0  alpha 10^-10  t < 10^-7 |
| Hare | F5 | Ones(5,1) | 6.4006e-08  6.4006e-08 | 1.0e-07 \*  -0.0131  0.1337  -0.0005  -0.0022  -0.0951 | 0.5259  0.0587 | 32  32 | 11  11 | 0.3293  0.1906  0.1906 | |  |
| Noll | F5 | Ones(5,1) | 0.0034  2.9962e-06  7.2611e-04  1.3843e-06  1.4839e-07 | 1.0e-03 \*  -0.3630  -0.0544  0.0798  -0.5521  0.0693 | 0.1906  0.5730  8.4961  0.1197  0.0560 | 49  66  2000  72  31 | 23  40  1986  21  11 | 0.0676  0.0143  0.0430 | Alpha  q > 10^6  alpha 🡪 10^-10  t < 10^-7  max iterations  t always so small  no warning | |

**Results table:**

* Algorithm seems fast but wrong
* Generally right direction! But too early stopping?
* Interesting that Hare always t\_min, Noll never 🡪 Hare more Null steps in the end, Noll more serious steps in the end? 🡪 just wrong factor  
  now higher t 🡪 more serious steps than Noll

**Results Blue:**

* Better result but still not good enough
* More steps than Hare now but time still very good

**Results Green:**

* Generally better optimal values
* 1 time more steps than with q
* 1 time not converged 🡪 in this one t really big all the time

**Results Red:**

* Worse 🡪 partly f, x good, but no stopping because Q too big (f1), partly f good but x really bad (f2) (how?), f3 ok, f4 like f1, f5 no stopping and not a good result
* (Sometimes) more steps

**Results Purple:**

* Good results, f really close to optimum but sometimes still not in tolerance (Criterion does not make this sure)
* More steps 🡪 in average smaller t 🡪 smaller step sizes
* Time faster than hare but probably because Hare uses quadprog

**Results orange:**

* Better results -> all below 10^-6 now
* Less steps
* Seems less warnings (no comment = no warnings)

**Results qpas:**

* Qpas 🡪 alpha schlechter (bis ca 10-09 Abweichung)
* A bit more steps with qpas
* Up to 10 times faster

**Testing/improvements**

* T\_min Hare, t Noll 🡪 check when null/serious steps are made
  + Not really clear why Noll higher t
  + Clear: different factors u2 🡪 write correct numbers into table
* Generally problem with alpha in Noll algo  
  🡪 !!!!! Hare Algorithm uses quadprog 🡪 use qpas
  + Qpas only for pos-semidefinite 🡪 check if Q+1/t\*I pos semi def (but if not res not finite???)
  + Done such that always pos semi def 🡪 !!!!!! have to find new lowest t\_min!!! to make sure t🡪 0 not possible
  + For f1 ok, for f2, f5 still problem 🡪 correction of t also needed in null steps
  + Problem: t becomes too small
* New idea: correct Q and not t
  + Correct with a factor  
    does this make sense? Theoretical background
  + Possible: correct only one eigenvalue -> how? Does Q-matrix make sense then?
* Check stopping condition, relation between d and G/S
* Other idea: find subgradients such that positive definiteness remains 🡪 how? Possible?

🡪 check when it happens in Noll algo (f1)

* + No comment in solution 🡪 alpha = 10^-11/12
  + comment 🡪 10^-01 (after a few steps)  
    Q has negative eigenvalue 🡪 this also the case if 10^-12 and no warning
  + bad alpha occurs when (Q+1/t\*I) indefinite! 🡪 case, that is not allowed
* Check when stopping condition fulfilled and why
* Hare: every time t\_min 🡪 why???
* Noll more serious steps??? 🡪 because t bigger 🡪 check

Changed: Q+1/t\*I is always pos semi def; not manipulating t but Q on its own

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algo | Test function | X0 | F | X | time | iter | Null steps | T end | Comments |
| Hare | F1 | Ones(5,1) | 2.0726e-07 | 1.0e-07 \*  0.4246  -0.2163  -0.3712  -0.2507  -0.1742 | 0.3234 | 20 | 6 | 0.2805 |  |
| Noll | F1 | Ones(5,1) |  |  |  |  |  |  |  |
| Hare | F2 | Ones(5,1) | 2.1955e-08 | 1.0e-04 \*  0.2237  0.2165  0.2098  0.2382  0.2157 | 0.6777 | 34 | 14 | 0.2617 |  |
| Noll | F2 | Ones(5,1) |  |  |  |  |  |  |  |
| Hare | F3 | Ones(5,1) | 2.2822e-08 | 1.0e-08 \*  0.8200  0.4651  -0.3206  -0.1358  0.8121 | 0.5741 | 33 | 9 | 0.8892 |  |
| Noll | F3 | Ones(5,1) |  |  |  |  |  |  |  |
| Hare | F4 | Ones(5,1) | 7.9240e-08 | 1.0e-07 \*  0.0728  -0.1077  0.1379  0.0425  -0.0720 | 0.4245 | 25 | 7 | 0.4653 |  |
| Noll | F4 | Ones(5,1) |  |  |  |  |  |  |  |
| Hare | F5 | Ones(5,1) | 6.4006e-08 | 1.0e-07 \*  -0.0131  0.1337  -0.0005  -0.0022  -0.0951 | 0.5259 | 32 | 11 | 0.3293 |  |
| Noll | F5 | Ones(5,1) | 2.9962e-06 | 1.0e-06 \*  -0.0636  -0.1846  0.0888  -0.3184  0.6020 | 0.5730 | 66 | 40 |  | q > 10^6  alpha 🡪 10^-10 |