Results_for_Trees_difference

August 28, 2023

1 Man benötigt das Oscar-Paket, um Algebra machen zu können. Das Laden dauert einige Zeit.

[1]:
```
using Pkg
Pkg.add("XLSX")
Pkg.add("Oscar")
import XLSX
using Oscar
```

```
    Updating registry at
`/scratch/htc/bzfweber/.julia/registries/General.toml`
   Resolving package versions…
  No Changes to
`/scratch/htc/bzfweber/.julia/environments/v1.8/Project.toml`
  No Changes to
`/scratch/htc/bzfweber/.julia/environments/v1.8/Manifest.toml`
   Resolving package versions…
  No Changes to
`/scratch/htc/bzfweber/.julia/environments/v1.8/Project.toml`
  No Changes to
`/scratch/htc/bzfweber/.julia/environments/v1.8/Manifest.toml`
 _____    _____    _____      _      _____
|     |  |     |  |     |    | |    |     |
|     |  |        |         |   |   |     |
|     |   _____   |        |     |  |_____
|     |        |  |        |_____|  |   |
|     |  |     |  |     |  |     |  |    |
 _____    _____    _____   _     _  _     _

…combining (and extending) ANTIC, GAP, Polymake and Singular
Version 0.11.2 …
 … which comes with absolutely no warranty whatsoever
Type: '?Oscar' for more information
(c) 2019-2022 by The OSCAR Development Team
```

2 Hier wird der Polynomring konstruiert, der die Eigenschaften ((w)et, (a)cidic, (n)eutral, (v)ery cold, (c)old und (g)rowing) der Beobachtungen zu "Auswahl-Statements" verknüpft.

[2]:
```
R1 = ResidueRing(ZZ, 2);
myR, (w,a,n,v,c,g) = PolynomialRing(R1, ["w", "a", "n", "v", "c", "g"]);
myvars=[w,a,n,v,c,g];
```

3 Die 18 Einzel-Beobachtungen, unter welchen Bedingungen der Baum wächst werden eingelesen.

[3]:
```
chartable=XLSX.readdata("growingTree.xlsx", "Sheet1", "A2:F19");
indices=1:18;
```

4 Anzahl der Eigenschaften und Auswahl der Eigenschaften

[4]:
```
varnum=6; choice=1:6;
```

5 Jede einzelne Beobachtung wird in ein Polynom übersetzt, das genau dem "Auswahl-Statement" entspricht, das diese Beobachtung aus allen herausselektiert.

[5]:
```
chartable=chartable[:,choice];
expr_list=prod(myvars+ myvars.^0 -chartable[1,:]);
for i=2:size(chartable,1)
    expr_list=vcat(expr_list, prod(myvars+ myvars.^0 -chartable[i,:]));
end
```

6 Die Eigenschaften bekommen ein Gewicht, das mit der ausgeglichenen Häufigkeit des Auftretens des Merkmals korreliert.

[6]:
```
monweight=ones(Int,varnum);
for i=1:size(myvars,1)
    print(myvars[i]);
    print(" true=");
    print(sum(chartable[:,i]));
    print(" false=");
    print(size(chartable,1)-sum(chartable[:,i]));
    print("\n");
    monweight[i]=sum(chartable[:,i])*(size(chartable,1)-sum(chartable[:,i]));
end
```

```
w true=9 false=9
a true=6 false=12
n true=6 false=12
v true=6 false=12
c true=12 false=6
g true=10 false=8
```

7 Die Menge der Nicht-Beobachtungen wird als Ideal generiert. Zudem wird die Idempotenz der Eigenschaften beachtet. Die Gröbnerbasis dieses Ideals wird berechnet und gibt eine Basis aller logischen Regeln wider, die in den Daten stecken.

[7]:
```
uq_expr_list=unique(expr_list);
expr=sum(uq_expr_list)+1;
generator=myvars.^2+myvars;
```

[8]:
```
II1=ideal(myR, vcat(generator, expr));
Y1, m1 = quo(myR, II1);
GB1 = groebner_basis(II1, ordering = wdeglex(gens(myR),monweight))
```

[8]:
```
Gröbner basis with elements
1 -> c^2 + c
2 -> v*c + v
3 -> v^2 + v
4 -> n^2 + n
5 -> a*n
6 -> a^2 + a
7 -> w*v + w*c + v*g + c*g + v + c
8 -> w*n + n*g + n*c
9 -> g^2 + g
10 -> w*g + w*c + w
11 -> w^2 + w
12 -> n*c*g + n*g + c*g + n*c + g + n + c + 1
13 -> a*c*g + a*g + a*c + a
14 -> a*v*g
15 -> w*a*c + a*v*g + w*c + a*g + c*g + a + c
with respect to the ordering
wdeglex([w, a, n, v, c, g], [81, 72, 72, 72, 72, 80])
```

8 Einen Teil dieser Regeln haben wir bereits erwartet. Wir sammeln die Nicht-Beobachtungen, die wir erwartet hätten. Draus wird wieder ein Ideal gemacht und eine Gröbner-Basis.

[9]:
```
expr2 = [v*(1+c), a*n];
```

[10]:
```
II2=ideal(myR, vcat(generator, expr2));
Y2, m2 = quo(myR, II2);
GB2 = groebner_basis(II2, ordering=wdeglex(gens(myR), monweight));
```

9 Die erwarteten und die gefunden Regeln werden zu einem Gesamtideal vereint. Wieder mit Gröbner-Basis.

[11]:
```
II3=ideal(myR, vcat(generator, expr, expr2));
Y3, m3 = quo(myR, II3);
GB3 = groebner_basis(II3, ordering=wdeglex(gens(myR), monweight));
```

10 Zu jeder Beobachtung wird (durch Division mit Reste bezüglich des Gesamtideals) festgestellt, welche Regel zusätzlich entstehen würde, würde man die spezielle Bobachtung streichen. (Das beschreibt die Besonderheit der entsprechenden Beobachtung)

[12]:
```
for i = 1:size(uq_expr_list,1)
    res=normal_form(uq_expr_list[i],gens(GB3));
    counter=0;
    for j = 1:size(expr_list,1)
        if (uq_expr_list[i]==expr_list[j])
            counter=counter+1;
            printstyled(indices[j], color=:red);
            print(" ");
        end
    end
    printstyled(" (#", color=:red);
    printstyled(counter, color=:red);
    printstyled(")", color=:red);
    print("\n");
    printstyled(factor(uq_expr_list[i]), color=:green);
    wrtnorm=0;
    if (wrtnorm==1)
        printstyled("=\n", color=:green);
        printstyled(uq_expr_list[i], color=:green);
    end
```

```
    print("\n");
    printstyled(res, bold=:true);
    print("\n\n");
end
```

```
1  (#1)
1 * g * (c + 1) * (v + 1) * (n + 1) * (a + 1) * (w + 1)
w*a*g + w*g + a*c + a + c*g + g

2  (#1)
1 * c * g * (v + 1) * (n + 1) * (a + 1) * (w + 1)
a*c + a*g + a + n*v*g + n*c + n*g + n + v*g + c + g + 1

3  (#1)
1 * v * c * g * (n + 1) * (a + 1) * (w + 1)
n*v*g + v*g

4  (#1)
1 * n * (g + 1) * (c + 1) * (v + 1) * (a + 1) * (w + 1)
c*g + c + g + 1

5  (#1)
1 * n * c * g * (v + 1) * (a + 1) * (w + 1)
n*v*g + n*c + n*g + n + c*g + c + g + 1

6  (#1)
1 * n * v * c * g * (a + 1) * (w + 1)
n*v*g

7  (#1)
1 * a * g * (c + 1) * (v + 1) * (n + 1) * (w + 1)
w*a*g + a*c + a

8  (#1)
1 * a * c * g * (v + 1) * (n + 1) * (w + 1)
a*c + a*g + a

9  (#1)
1 * a * v * c * (g + 1) * (n + 1) * (w + 1)
w*a*g + w*a + a*g + a

10  (#1)
1 * w * g * (c + 1) * (v + 1) * (n + 1) * (a + 1)
w*a*g + w*g + n*c + n + c*g + c + g + 1

11  (#1)
1 * w * c * (g + 1) * (v + 1) * (n + 1) * (a + 1)
```

```
a*v + a*g + a + n*v*g + n*v + n*g + n + v*g + v + g + 1
```

```
12  (#1)
1 * w * v * c * (g + 1) * (n + 1) * (a + 1)
w*a*g + w*a + w*g + w + a*v + a*g + a + n*v*g + n*v + v*g + v + c*g +
c
```

```
13  (#1)
1 * w * n * g * (c + 1) * (v + 1) * (a + 1)
n*c + n + c*g + c + g + 1
```

```
14  (#1)
1 * w * n * c * (g + 1) * (v + 1) * (a + 1)
n*v*g + n*v + n*g + n + c*g + c + g + 1
```

```
15  (#1)
1 * w * n * v * c * (g + 1) * (a + 1)
n*v*g + n*v
```

```
16  (#1)
1 * w * a * g * (c + 1) * (v + 1) * (n + 1)
w*a*g
```

```
17  (#1)
1 * w * a * c * (g + 1) * (v + 1) * (n + 1)
a*v + a*g + a
```

```
18  (#1)
1 * w * a * v * c * (g + 1) * (n + 1)
w*a*g + w*a + a*v + a*g + a
```

11 Wieder Division mit Rest: Es wird geschaut, welche Regeln aus den Beobachtungen übrig bleiben, wenn man die Erwartungen "rauskürzt".

[13]:
```
for i=1:length(GB1)
    nf=normal_form(GB1[i], gens(GB2));
    if (nf!=0)
        print(i); print(" -> ");
        print(nf);
        print("\n");
    end
end
```

```
7 -> w*v + w*c + v*g + v + c*g + c
```

```
8 -> w*n + n*c + n*g
10 -> w*c + w*g + w
12 -> n*c*g + n*c + n*g + n + c*g + c + g + 1
13 -> a*c*g + a*c + a*g + a
14 -> a*v*g
15 -> w*a*c + w*c + a*v*g + a*g + a + c*g + c
```

[ ]: