Optimal Weights and Activation Values

Data: In Sanskrit we observe an alternation between several allomorphic roots for one and the same verb. Let's take *budh* 'awake, know, ...' as a concrete example

- bodh-ati (3sg. pres.)
- bhot-sya-ti (3sg. fut.)
- bhut (Noun, Nom.)
- bhud-bhis (Noun, Instr.)
- bud-dha (Past participle)

GSC offers a possible solution to this puzzle if we assume that the input for morphological derivations is a blend of different allomorphs with some partial activation values (see GSCNet).

But how do we find suitable values for these activations and how do we find the right constraint weights in a Gradient Harmonic Grammar?

The problem can be phrased as a mathematical optimization problem: We have a function:

$$H_{winner}(x) = c^T x$$

expressing the Harmony (sum of violations) of the winner, that we want to maximize. Our xs are the Harmonic constraints and the coefficients c are the violations for the winning candidate. This expression is subject to some constraints: for each other candidate it should hold that

$$H_{winner} > H_{candidate_i}$$

$$\forall cand_i \in Tableaux$$

We also want the weights to be negative (these are penalties):

$$[x_i, x_2, x_3 \ldots x_n] < 0$$

and we know that the activation values are between 0 and 1: 0 meaning inactive and 1 meaning totally active (discrete).

$$1 \leq [\alpha \ldots . \omega] \geq 0$$

Example: the past participle

Inputs: root allomorphs:

- (bhud)$^{\alpha}$
- (budh)$^{\beta}$
- (bhut)$^{\gamma}$

suffix allomorphs:

- (-ta)$^{\delta}$
- (-dha)$^{\epsilon}$

Constraints: (all of them < 0)

- Max : anti-deletion constraint
- Dep : anti-epenthesis constraint
- Voice : Avoid voice contrast ([+ voice][-voice] is bad)
- License(lar) : Aspirated segments cannot occur before voiceless segments. Also avoid two aspirated next to each other.
- Lazy : Reduce articulatory effort (avoid aspirated sounds)
- *h]_W:Aspirated are not allowed in word final position
- Id(lar) : Aspirated in the input == Aspirated in the output

Toy-example

$root_{(\alpha+\beta+\gamma)} + suff_{(\delta+\epsilon)}$	Max	Dep	Voice	Lic	h]W	Lazy	Id(Lar)	Harmony
bhud-ta	$\beta+\gamma+\epsilon$	$(2-\alpha-\delta)$	1	0	0	0	0	$M(\beta+\gamma+\epsilon)+D(2-\alpha-\delta)+Voice$
:) bud-dha	$\alpha+\gamma+\delta$	$(2-\beta-\epsilon)$	0	0	0	0	1	$M(\alpha+\gamma+\epsilon)+D(2-\beta-\epsilon)+Id(lar)$
budh-i-ta	$\alpha+\gamma+\epsilon$	$(3-\beta-\delta)$	0	0	0	0	1	$M(\alpha+\gamma+\epsilon)+D(3-\beta-\delta)$
...	...	...	...	...	...	...	...	...

Notice that candidate 3 (*budh-i-ta*, built on the model of *kridh-i-ta*) has an extra epenthesis, hence the violation for dep (**3** - $\beta - \delta$)

Let's now solve this problem using the *JuMP* library for mathematical optimization. The expression we want to maximize is the Harmony of the winner (candidate 2):

$$max : M(\alpha+\gamma+\epsilon)+D(2-\beta-\epsilon)+Id(lar)$$

subject to:

$$M(\alpha+\gamma+\epsilon)+D(2-\beta-\epsilon)+Id(lar) > M(\beta+\gamma+\epsilon)+D(2-\alpha-\delta)+Voice$$

$$M(\alpha+\gamma+\epsilon)+D(2-\beta-\epsilon)+Id(lar) > M(\alpha+\gamma+\epsilon)+D(3-\beta-\delta)$$

$$0 < [\alpha, \beta, \gamma, \delta, \epsilon] < 1$$

$$M, D, Id(lar), Voice, Lic, h]W < 0$$

\text{$$ \begin{alignat*}{1}\text{feasibility}\\ \text{Subject to} \quad\end{alignat*} $$}

```
# Initialize Model
begin
    optimizer = Juniper.Optimizer
    nl_solver = optimizer_with_attributes(Ipopt.Optimizer, "print_level" => 0)
    model = Model(optimizer_with_attributes(optimizer, "nl_solver" => nl_solver))
end
```

```

```

```
-1
```

```
begin
    # Declare variables
    a = 1 # The difference between winner and losers should be greater than a
    b = -1 # Start value for the constraint weights
end
```

ϵ

```
begin
    @variable(model,  -10 <= Ma <= -1, start = b,integer = true) # MaxIO
    @variable(model,  -10 <= De <= -1, start = b,integer = true) # DepIO
    @variable(model,  -10 <= V <= -1, start = b,integer = true) # Assimilate == Voice
    @variable(model,  -10 <= Lic <= -1, start = b,integer = true) # License(lar)
    @variable(model, -10 <= hW <= -1, start = b,integer = true) # h]W
    @variable(model, -10 <= La <= -1, start = b,integer = true) #Lazy
    @variable(model, -10 <= IdL <= -3, start = 2 * b,integer = true) # Ident(lar)
    @variable(model, 0.1 <= α <= 0.9, start = 0.2) # bhud
    @variable(model, 0.1 <= β <= 0.9, start = 0.9) # budh
    @variable(model, 0.1 <= γ <= 0.9, start = 0.4) # bhut
    @variable(model, 0.1 <= δ <= 0.5, start = 0.5) # dha
    @variable(model, 0.5 <= ϵ <= 0.9, start = 0.7) # ta
end
```

\text{$$ Ma\times α + Ma\times γ + Ma\times δ - β\times De - ϵ\times De + 2 De + IdL $$}

```
@objective(model, Max, Ma * (α + γ + δ) + De * (2 - β - ϵ) + IdL)
```

```
"@objective(model, Max, Ma * (α + γ + δ) + De * (2 - β - ϵ) + IdL)"
```

```
# Objective function : Harmony of the winner
"""@objective(model, Max, Ma * (α + γ + δ) + De * (2 - β - ϵ) + IdL)"""
```

$$-De + IdL \geq 1.0$$

```
# Constraints
begin
    # Winner Harmony
    winner = @expression(model, Ma * (α + γ + δ) + De * (2 - β - ϵ) + IdL)

    bhudta = @expression(model, winner - (Ma * (β + γ + δ) + De * (2 - α - ϵ) + V))
    @constraint(model, bhudta >= a)

    budhta = @expression(model, winner - (Ma * (α + γ + δ) + De * (2 - β - ϵ) + V + Lic))
    @constraint(model, budhta >= a)

    budhita = @expression(model, winner - (Ma * (α + γ + δ) + De * (3 - β - ϵ)))
    @constraint(model, budhita >= a)
end
```

```
"Max Ma*α + Ma*γ + Ma*δ - β*De - ϵ*De + 2 De + IdL\nSubject to\n Ma*α - β*Ma + De*α - β*De - V + IdL >= 1.0\n -V - Lic + Id
```

```
"""$model"""
```

```
# Optimize / Solve
optimize!(model)
```

```
-4.09999984904502
```

```
# Printing logic
# Objective value (Harmony of the winner after the optimization)
JuMP.objective_value(model)
```

```
FEASIBLE_POINT::ResultStatusCode = 1
```

```
# Solver status
JuMP.primal_status(model)
```

```
LOCALLY_SOLVED::TerminationStatusCode = 4
```

```
JuMP.termination_status(model)
```

```
-1.0
```

```
# MaxIO
value(Ma)
```

```
-3.99999996003281
```

```
#Dep
value(De)
```

```
-10.0
```

```
# Voice
value(V)
```

```
-9.000000009999997
```

```
# Coda
value(hW)
```

```
0.1
```

```
value(α)
```

```
0.9
```

```
value(β)
```

```
0.1
```

```
value(γ)
```

```
0.1
```

```
value(δ)
```

```
0.9
```

```
value(ϵ)
```

Of course this was only a toy-example. JuMP evaluated the expressions for just two candidates and some value were arbitrarly fixed but I think it gives the idea, how Mathematical Optimization (Quadratic Programming in this case) can be used to search suitable values for the constraints and the activation values