# No title

Erik Hartman[1] Jonas Wallin[2]

[1] Division of Infection Medicine, Faculty of Medicine, Lund University
[2] Department of Statistics, Lund University

## Algorithms

**Algorithm 1:** Simulate proteolysis of protein $P$.

**Input:** protein $P$, $n_{generate}$ $\theta_{enzyme}$, $\theta_{gamma}$, $p_{endo}$, $p_{exo}$
**Output:** $f$

$f \leftarrow$ dict[sequence : count]
$f(P) \leftarrow n_P$
**while** $i < n_{generate}$ **do**
  $n_{generate} \leftarrow \sum_x T(x)$
  $x \sim U(0,1)$
  **if** $x > p_{endo}$ **then**
    //exoprotease
    sequence to chew $s \sim f$ //sample sequence from dict with weights count
    $x \sim U(0,1)$
    $a \leftarrow gamma(len(s), \theta_{gamma})$
    **if** $a < x$ **then**
      //accept
      $F(P) \leftarrow F(P) + 1$
      $n_{generate} \leftarrow n_{generate} + 1$
    **end**
  **else**
    //endoprotease
    $f_{cut}(s) \leftarrow \sum_s N^s_{aa} * \theta_{aa}$
    sequence to cut $s \leftarrow f_{cut}(s)$//sequence to cut
    first index to cut $index_a \sim s(\theta_{aa}(aa_x))$
    $index_2 \sim s(\theta_{aa}(aa_x)) * gamma(|index_1 - index_2|)$
    left $\leftarrow s[: min(index_1, index_2) + 1]$
    middle $\leftarrow s[min(index_1, index_2) + 1 : max(index_1, index_2)]$
    right $\leftarrow s[max(index_1, index_2) + 1 :]$
    **if** *len(middle)>5* **then**
      $f(middle) \leftarrow f(middle) + 1$
      $n_{generate} \leftarrow n_{generate} + 1$
    **end**
    **for** *s in [left, right]* **do**
      $x \sim U(0,1)$
      $a \leftarrow gamma(len(s), \theta_{gamma})$
      **if** $a < x$ *and s is not terminal peptide in* $P$ **then**
        //accept
        $F(P) \leftarrow F(P) + 1$
        $n_{generate} \leftarrow n_{generate} + 1$
      **end**
    **end**
  **end**
**end**
Return $f$

**Algorithm 2:** Estimating $\theta$ numerically. To generate a guess, simulate degradation of protein $P$ with parameters $\theta$ to generate $n_{generate}$ peptides (see Algorithm 1).

**Input:** protein $P$, $n_P$, true distribution $T$, $\theta$, $lr_{endo}$, $lr_{exo}$
**Output:** $\theta$

**for** *i from 0 to* $n_{endo}$ **do**
  Generate guess $G$;
  Compute loss $L \leftarrow D_{KL}(G||T) + D_{KL}(T||G)$;
  **for** *each amino acid aa in* $\theta_{aa}$ **do**
    $\theta_{aa}(aa) \leftarrow \theta_{aa}(aa) + lr_{endo}$;
    Generate guess $\hat{G}$ with new $\theta$;
    Compute new loss $\hat{L} \leftarrow D_{KL}(\hat{G}||T) + D_{KL}(T||\hat{G})$;
    **while** $\hat{L} < L$ **do**
      Compute weighted learning rate $lr_w \leftarrow lr_{endo} * \hat{L} - L$;
      $L \leftarrow \hat{L}$;
      $\theta_{aa}(aa) \leftarrow \theta_{aa}(aa) + lr_w$;
      Generate guess $\hat{G}$ with new $\theta$;
      Compute new loss $\hat{L} \leftarrow D_{KL}(\hat{G}||T) + D_{KL}(T||\hat{G})$;
    **end**
    $\theta_{aa}(aa) \leftarrow \theta_{aa}(aa) - lr_{endo}$ //revert the initial parameter-change (before while-loop);
  **end**
**end**
Generate guess $G$;
Compute loss $L \leftarrow D_{KL}(G||T) + D_{KL}(T||G)$;
**for** *i from 0 to* $n_{exo}$ **do**
  $x \leftarrow$ uniformly random from $1, -1$;
  $e \leftarrow lr_{exo} * x$;
  $\theta_{exo} \leftarrow \theta_{exo} + e$;
  Generate guess $\hat{G}$ with new $\theta$;
  Compute new loss $\hat{L} \leftarrow D_{KL}(\hat{G}||T) + D_{KL}(T||\hat{G})$;
  **if** $\hat{L} > L$ **then**
    $\theta_{exo} \leftarrow \theta_{exo} - e$;
  **else**
    $L \leftarrow \hat{L}$;
  **end**
**end**
Return $\theta$;

**Algorithm 3:** Estimate weights in graph with gradient descent.

**Input:** observed distribution $T$, $lr$, $\lambda_1$, $\lambda_2$, $n_{iterations}$

**Output:** $G$

$G \leftarrow \{V, E, W\}$

$V_L$ denotes the original protein (bottom node)

**for** *i from 0 to* $n_{iterations}$ **do**

  //generate guess

  $p_{generated} \leftarrow \{\}$// this represents the output distribution if starting from a given node

  $T = \{s \in V \mid$ there exists no $(s,t) \in E$ for any $t \in V\}$ //terminal nodes

  **for** $t \in T$ **do**

    $p_{generated}[t] \leftarrow \mathbf{1}_t$ //onehot

  **end**

  **while** *all nodes in* $G$ *is not solved* **do**

    solvable $\leftarrow \{s \in V \mid t \in p_{generated}$ for all $(s,t) \in E\}$

    **for** $s \in$ *solvable* **do**

      $p_{generated}[s] = \sum w_{s,t} * p_{generated}[t] + 1 - \sum w_{s,t} * \mathbf{1}_t$

    **end**

  **end**

  $\hat{T} \leftarrow p_{generated}[V_L]$

  //compute loss

  $L_1 \leftarrow \lambda_1 \sum |w|$

  $L_2 \leftarrow \lambda_2 \sum w^2$

  $L \leftarrow D_{KL}(T \mid \hat{T}) + D_{KL}(\hat{T} \mid T) + L_1 + L_2$

  //compute gradient

  $\frac{dT}{dw} \leftarrow \hat{T}_{V_L,t}(\hat{T}_s - \mathbf{1}_s)$

  $\frac{dL}{dT} \leftarrow -\frac{T}{\hat{T}}$

  $\frac{dL}{dw} \leftarrow \frac{dL}{dT} * \frac{dT}{dw}$

  //update graph

  $k \leftarrow 1$

  **for** $s \in V$ **do**

    $\hat{w}_{s,t} \leftarrow max(0, w_{s,t} - lr * (\frac{dL}{dw})_{s,t})$

    $d_{s,t} = \hat{w}_{s,t} - w_{s,t}$

  **end**

  **while** $\sum_t w_{s,t} + k * d_{s,t} > 1$ **do**

    $k \leftarrow k/2$

  **end**

  **while** *a better graph is not found or* $k$ *isn't extremely small* **do**

    $\hat{W} \leftarrow W + d * k$

    $\hat{T} \leftarrow$ generate guess with new weights

    $\hat{L} \leftarrow D_{KL}(T \mid \hat{T}) + D_{KL}(\hat{T} \mid T) + L_1 + L_2$

    **if** $\hat{L} \leq L$ **then**

      $G \leftarrow \{V, E, \hat{W}\}$

    **else**

      $k \leftarrow k/2$

    **end**

  **end**

**end**

Return $G$