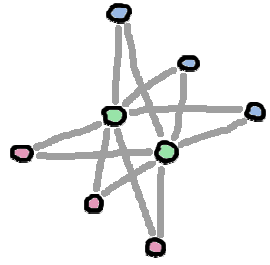
Rekurrentsed kihilised neurovõrgud ja pikk lühiajaline mälu

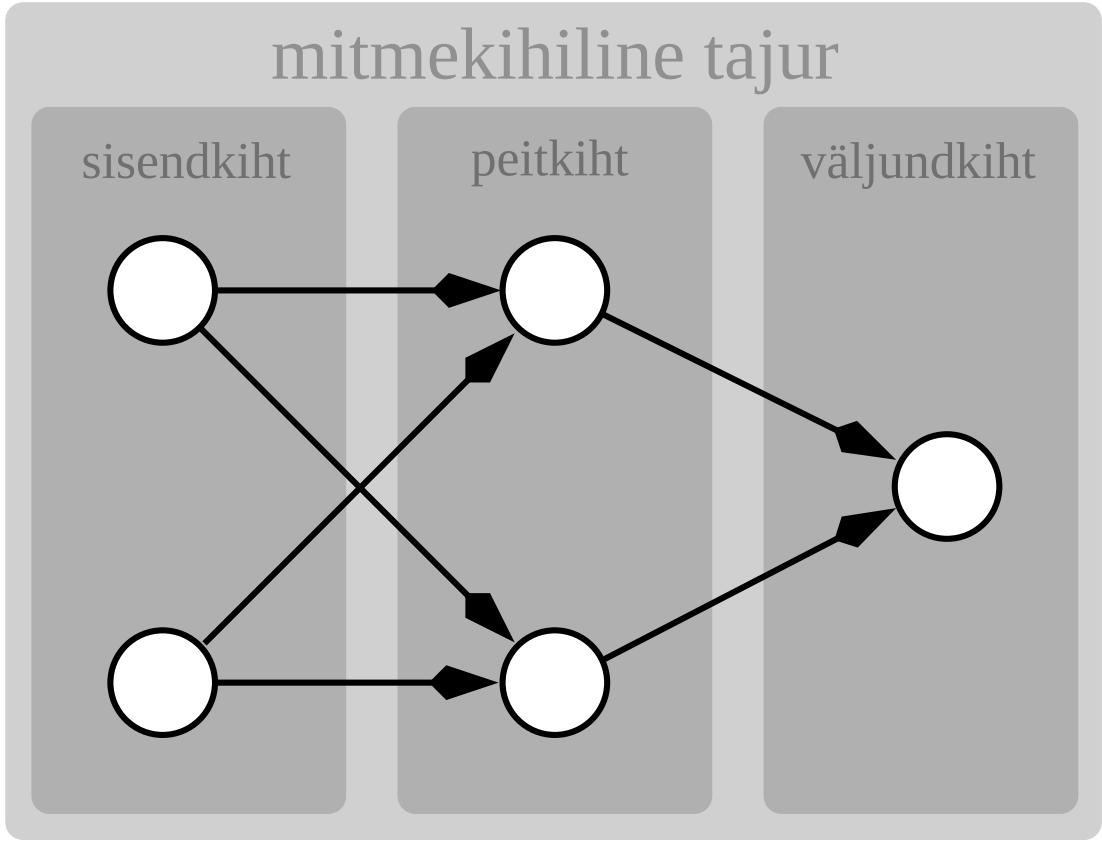
Mark Fišel (fishel@ut.ee)

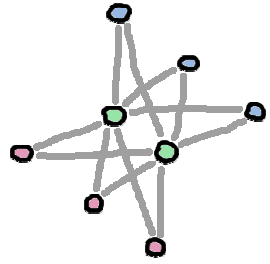
4. detsember 2006



MLP

Võtame aluseks mitmekihilist tajurit:



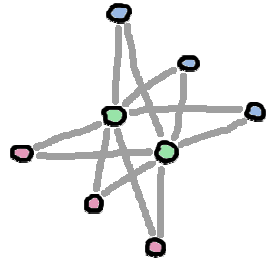


MLP

MLP väljundi arvutamine:

$$\begin{aligned} v_j &= \sum_{i \in \mathcal{I}_j} w_{ji} y_i, \\ y_j &= \varphi(v_j) \end{aligned}$$

kus $\mathcal{I}_j$ on neuroni j sisendneuronite hulk,
v_j – kohalik väli,
y_j – väljundväärtus

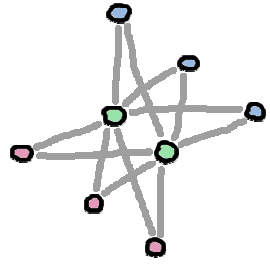


Diskreetne aeg

Lisame diskreetse aja mõiste:

$$\begin{aligned} v_j(t) &= \sum_{i \in \mathcal{I}_j} w_{ji} y_i(t), \\ y_j(t) &= \varphi(v_j(t)), \end{aligned}$$

kus $t \in \mathbf{N} \cup \{0\}$



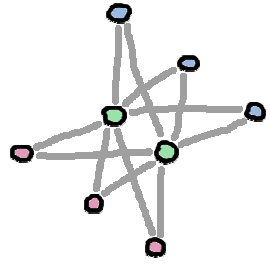
Viivitatud ühendused

Def: viivitatud ühendus – selline ühendus $j \longrightarrow i$, kus neuroni j sisendiks hetkel t on neuroni i väljund hetkel $t-1$.

Uus kohaliku välja valem:

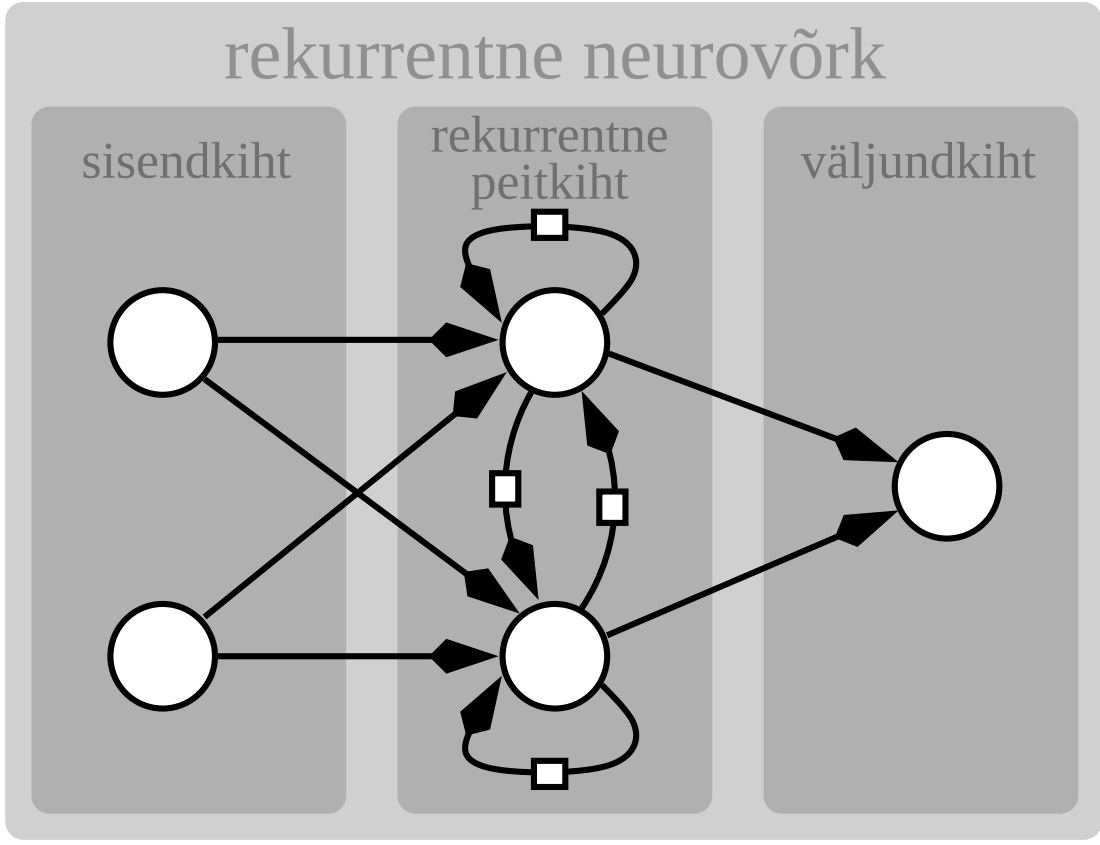
$$v_j(t) \;=\; \sum_{i \in \mathcal{I}_j} w_{ji} y_i(t - \Delta_{ji}),$$

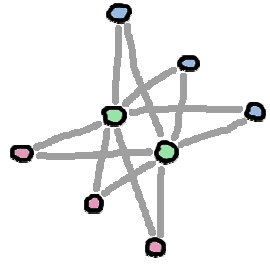
kus $\Delta_{ji} \in \{0, 1\}$
(vastavalt, tavaline või viivitatud ühendus)



RNN

Rekurrentne kihiline neurovõrk:





Tulemus

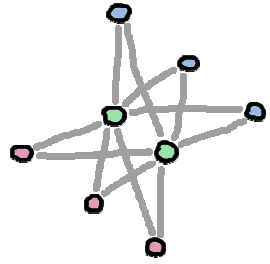
Mida head see annab?
MLP:

$$\mathbf{y} = f(\mathbf{x}, \mathbf{W})$$

RNN:

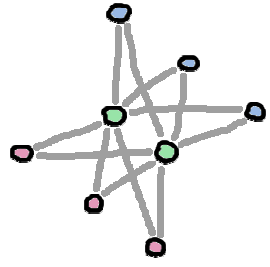
$$\begin{aligned} \mathbf{y}(t) &= f(\mathbf{x}(t), \mathbf{y}(t-1), \mathbf{W}) \\ &= g(\mathbf{x}(t), \mathbf{x}(t-1), \mathbf{y}(t-2), \mathbf{W}) \\ &= \cdots = h(\mathbf{X}, \mathbf{W}), \end{aligned}$$

$$\text{kus} \quad \mathbf{X} = \{\mathbf{x}(t), \mathbf{x}(t-1), \ldots, \mathbf{x}(0)\}$$



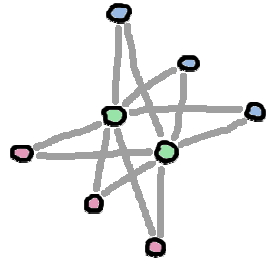
RNN treenimise variandid

- epohhipõhine: sisend-väljund järjendid on lõplikud, treenitakse peale ühe või mitu epohhi esitamist võrgule
- pidev: sisend-väljund järjendite suurust pole teada (potentsiaalselt lõpmatud), treenitakse peale igat ajasammu



RNN treenimise algoritmid

- BPTT: Vea tagasilevitamine läbi aega (ingl. *back-propagation through time*
- tBPTT: Kärbitud (ingl. *truncated*) vea tagasilevitamine läbi aega
- RTRL: Reaalajaline rekuurentne õppimine (ingl. *real-time recurrent learning*)



BPTT

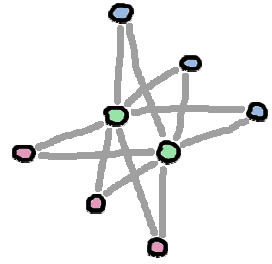
Defineerime

väljundneuroni viga: $e_j(t) = d_j(t) - y_j(t),$

ajahetke t viga: $\mathcal{E}(t) = \frac{1}{2} \sum_{j \in \mathcal{O}} (e_j(t))^2$

ja terve järjendi viga: $\mathcal{E} = \sum_t \mathcal{E}(t)$

Minimeerides $\mathcal{E}$ minimeerime ka iga $e_j(t)$.



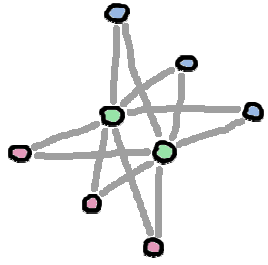
BPTT

Minimeerida saab nt. langeva gradiendi meetodi abil:

$$w_{ji} += -\eta \frac{\partial \mathcal{E}}{\partial w_{ji}} = \eta \sum_{t=0}^{t_{fin}} y_i(t - \Delta_{ji}) \cdot \delta_j(t),$$

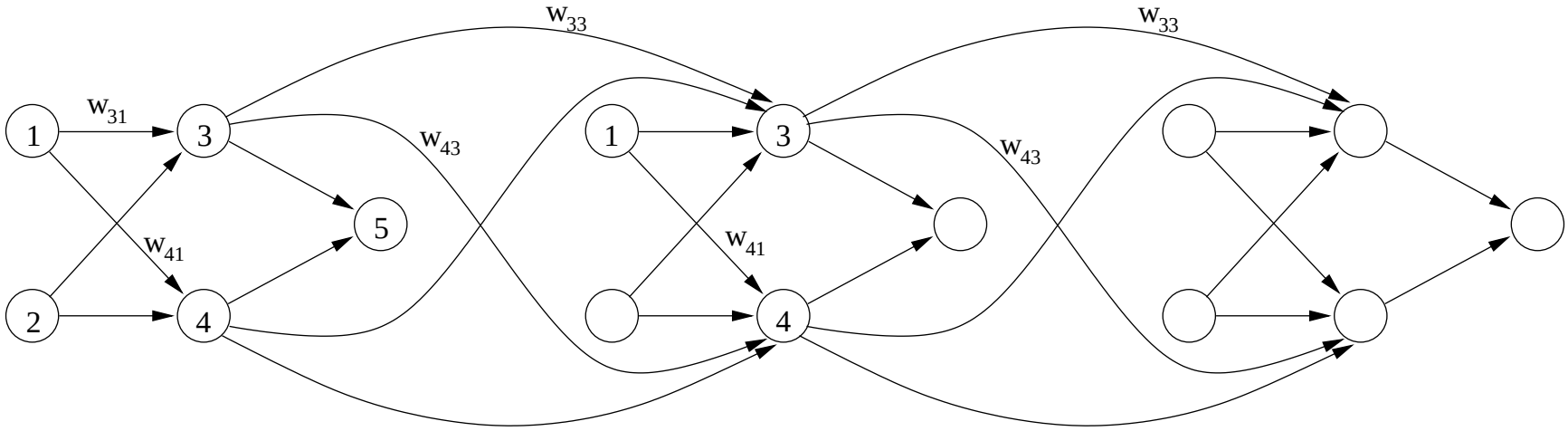
kus

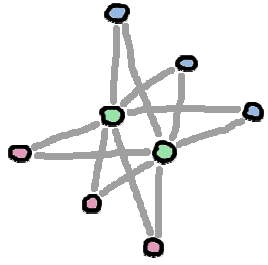
$$\delta_j(t) = \varphi'(v_j(t)) \cdot \sum_{k \in \mathcal{O}_j} w_{kj} \cdot \delta_k(t + \Delta_{kj})$$



Lahti rullimine

BPTT on nagu BP, kui võrk *lahti rullida*:

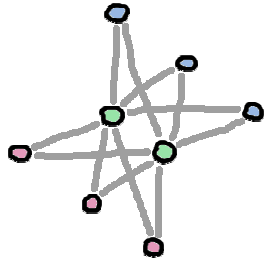




tBPTT

Kui sisendjärjend võib olla lõpmatu, ei saa summeerida 0st t_n'ni. Selle asemel kärbitud algoritmi variandis summeeritakse üle $k+1$ viimast elementi:

$$w_{ji} \mathrel{+}= \eta \sum_{t=t_{fin}-k}^{t_{fin}} y_i(t - \Delta_{ji}) \cdot \delta_j(t),$$



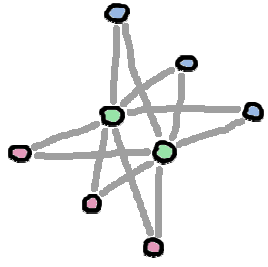
RTRL

RTRL algoritm vältib kärmist:

$$\frac{\partial \mathcal{E}(t)}{\partial w_{ji}} = \sum_{k \in \mathcal{O}} \frac{\partial \mathcal{E}(t)}{\partial e_k(t)} \frac{\partial e_k(t)}{\partial y_k(t)} \frac{\partial y_k(t)}{\partial w_{ji}} = -\sum_{k \in \mathcal{O}} e_k(t) \frac{\partial y_k(t)}{\partial w_{ji}},$$

ja nüüd avaldame $y_k(t)$ osatuletise rekursiivselt:

$$\frac{\partial y_k(t)}{\partial w_{ji}} = \varphi'(v_k(t)) \cdot \left(\delta_{kj} y_i(t - \Delta_{ki}) + \sum_p w_{kp} \cdot \frac{\partial y_k(t - \Delta_{kp})}{\partial w_{ji}} \right)$$

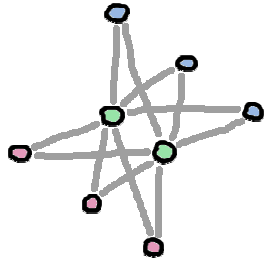


RTRL

alguses

$$\frac{\partial y_k(0)}{\partial w_{ji}} = 0$$

aega t suurenemisel arvutame $y_k(t)$ osatuletise rekursiivselt kasutades valemit ning selle abil kohendame kaalud nagu BPTT algoritmis

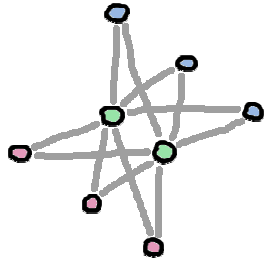


RNN lühike mälu

Tegelikkuses ei kehti rekurrentsete võrkude puhul

$$\mathbf{y}(t) = F(\mathbf{X}, \mathbf{W}),$$

sest mida suurem on t, seda nõrgem on seos "vanemate" sisenditega – kaduva gradiendi probleemi tõttu:



Kaduv gradient

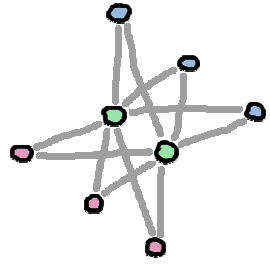
Gradient skaleerub q ajasammude jooksul suurusega:

$$\frac{\partial \delta_k(t-q)}{\partial \delta_p(t)} = \sum_{l_1} \cdots \sum_{l_{q-1}} \prod_{m=1}^{q} \varphi'(v_{l_m}(t-m)) \cdot w_{l_m l_{m-1}}$$

Korrutis summade all plahvatab või kahaneb eksponentsiaalselt q suhtes, kui

$$\left|\varphi'(v_{l_m}(t-m)) \cdot w_{l_m l_{m-1}}\right|$$

on vastavalt > 1 või < 1.



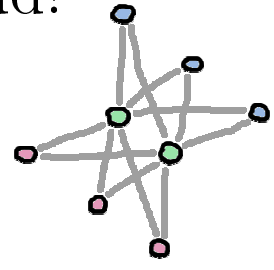
Kaduv gradient

Logistilise φ puhul φ' maksimum on $\frac{1}{4}$;

- kui $|w_{l_m l_{m-1}}| < 4$, korrutis kahaneb,
- kui $|w_{l_m l_{m-1}}| \to \infty$, korrutise liige läheneb 0'le

Järeldus: kui järjend on pikk, rekurrentne võrk kas ei õpi (gradient plahvatab), või ei mäleta väga palju oma minevikust (gradient kahaneb).

Praktikas võrk ei ole suuteline õppida sõltuvusi sisendite vahel, mis on 7–8 ajasammuga eraldatud!



Konstantse vea karusell

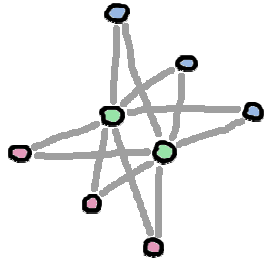
Lahendus: arvestame kaduva gradiendiga võrgu disainimisel; piirdume ühe neuroniga j ja sunnime konstantse veavoogu läbi selle:

$$\varphi'(v_j(t)) \cdot w_{jj} = 1$$

Siit automaatselt järeldub

$$\begin{aligned} \varphi(x) &= x, \\ w_{jj} &= 1 \end{aligned}$$

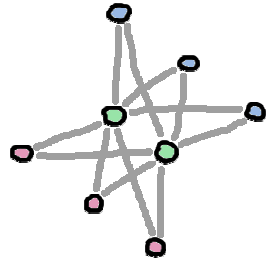
Selle konstruktsiooni nimetame konstantse vea karuselliks (constant error carousel, CEC)



Sisend- ja väljundväravad

Aga kui j on ka teiste neuronitega ühendatud tekkivad probleemid:

- võrgu väljundi arvutamisel peab neuroni sisendsignaaliga kord arvestama, kord mitte
- kui seda üritada teha tavalise ühenduse abil, selle kaal saab vasturääkivalt kohendatud
- sama lugu väljundist tuleva vea signaaliga

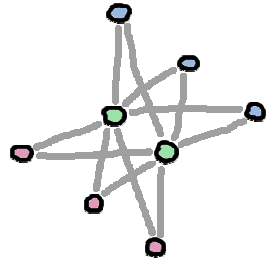


Sisend- ja väljundväravad

Lahendus: sisend- ja väljundväravad:

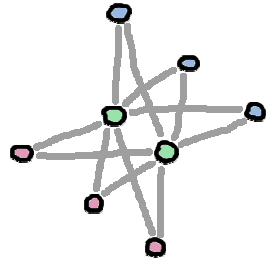
- väravad – tavalised neuronid
- neuroni sisend- ja väljundsignaali korrutatakse vastavate väravate väljundiga

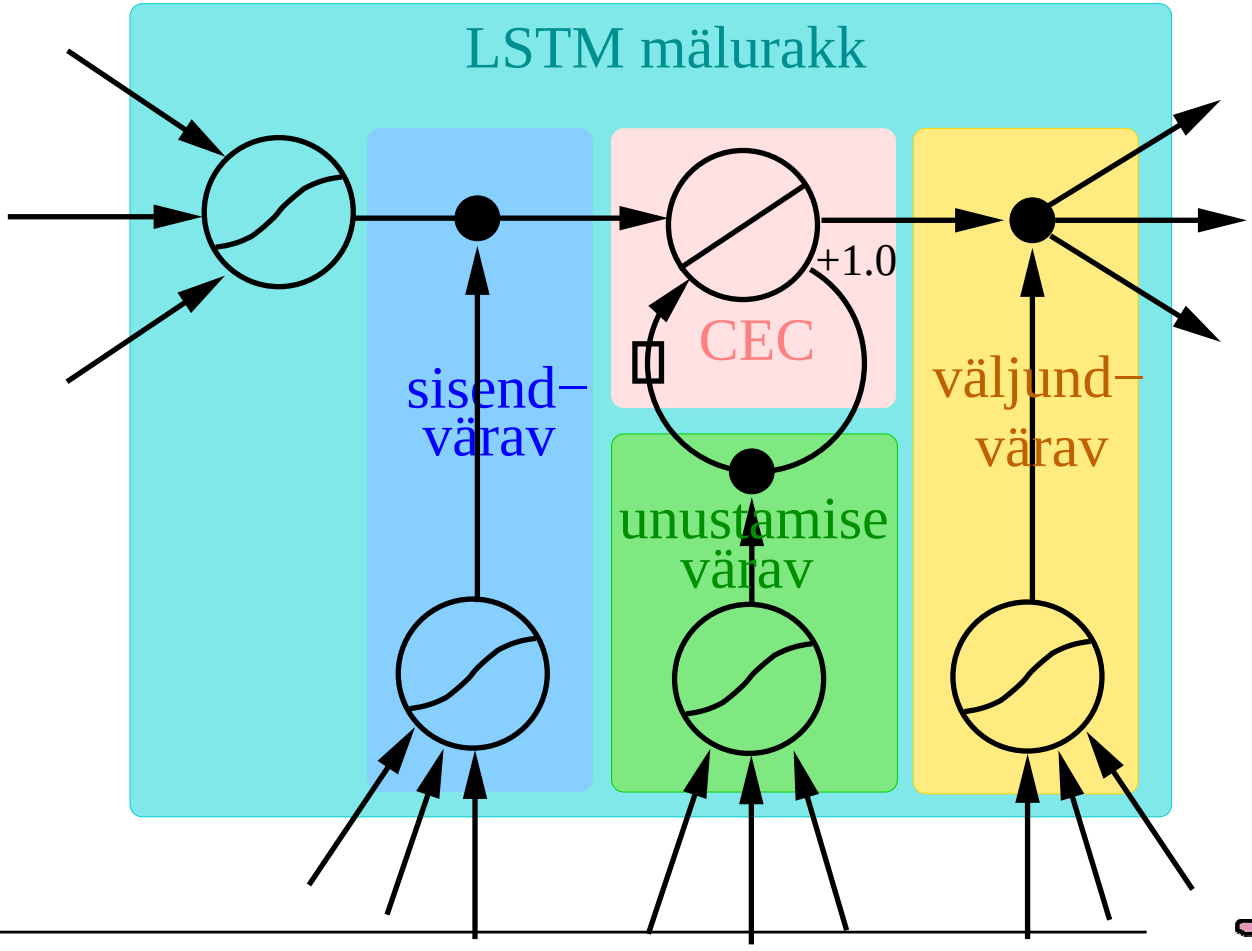
Neuronit, kus on CEC ja väravad, nimetame LSTM mälurakuks.

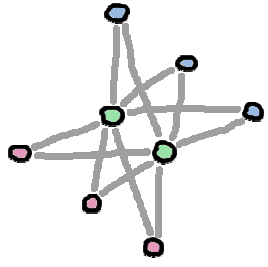


Unustamise värav

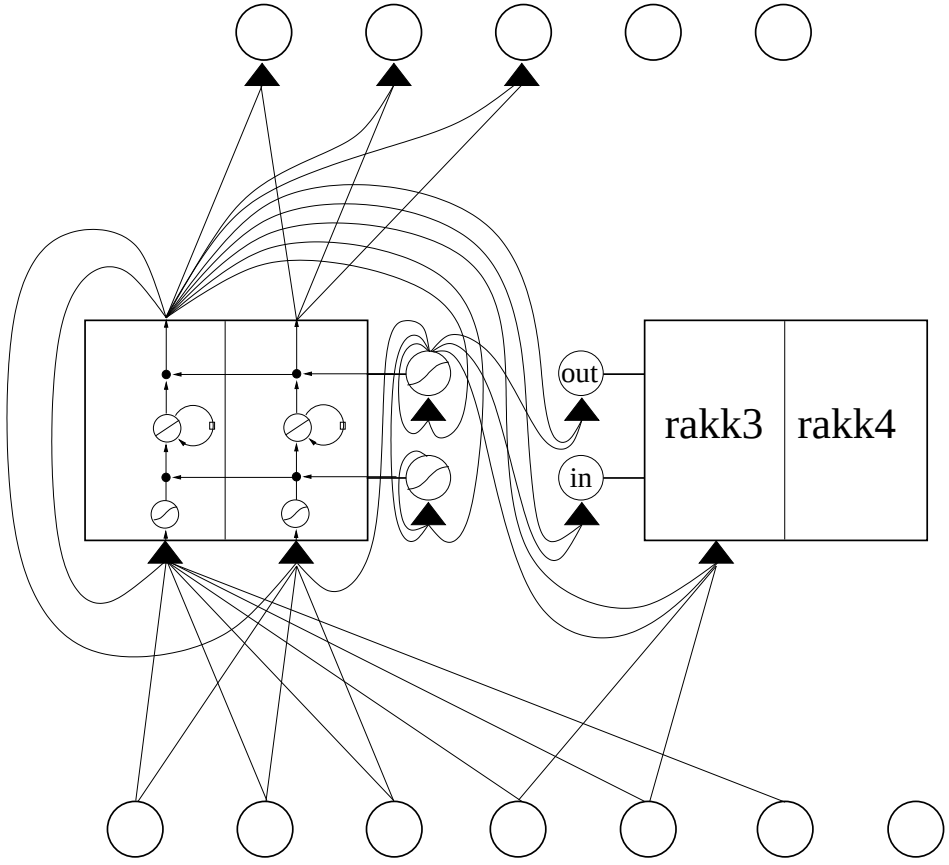
- osutub et LSTM rakk mäletab liiga hästi, ja mõnikord on vaja “unustada” praegust seisundit
- lisame unustamise värava, mille väljundiga korrutatakse eelmist seisundit

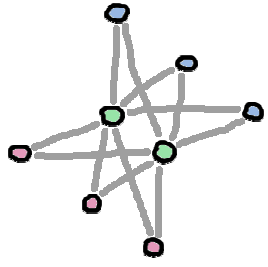



LSTM mälurakk
+1.0
CEC
sisend−
värav
unustamise
värav
väljund−
värav



Näidis konfiguratsioonist

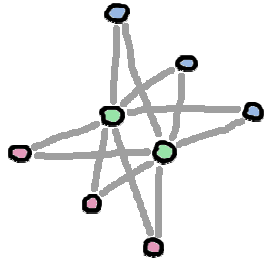




Tulemus

LSTM võrk oskab

- õppida sõltuvusi sisendite vahel, mis on eraldatud pika ajaga
- immiteerida pendlit
- liita kokku arvud
- töötada heliloojana
- ja veel palju kasulikke asju



LSTM raku väljund

lokaalne väli:

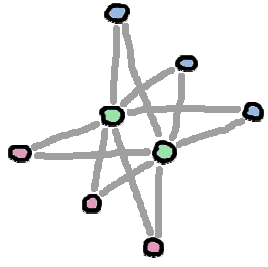
$$v_{c_j^v}(t) = \sum_i w_{c_j^v i}\, y_i(t - \Delta_{c_j^v i})$$

CEC seisund:

$$s_{c_j^v}(t) = s_{c_j^v}(t-1)\, y_{c_j^{fgt}}(t) + g(v_{c_j^v}(t))\, y_{c_j^{in}}(t)$$

raku väljund:

$$y_{c_j^v}(t) = h(s_{c_v^t}(t))\, y_{c_j^{out}}(t)$$



LSTM treenimine

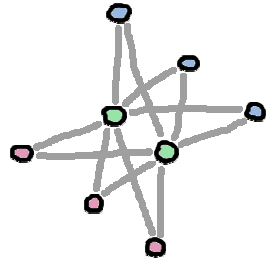
Treenitakse RTRL-i sarnase algoritmiga:

- hoitakse mälus osatuletised

$$\frac{\partial y_{c_j^v}(t)}{\partial w_{c_j^v i}}, \quad \frac{\partial y_{c_j^v}(t)}{\partial w_{c_j^{in} i}}, \quad \frac{\partial y_{c_j^v}(t)}{\partial w_{c_j^{fgt} i}} \quad \text{ja} \quad \frac{\partial y_{c_j^v}(t)}{\partial w_{c_j^{out} i}}$$

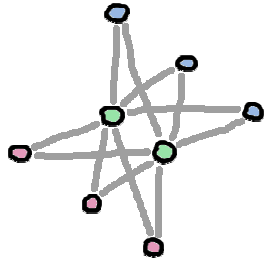
- nende abil arvutatakse üldise vea tuletise ning kohendatakse kaalud:

$$w_{c_j^v i} -= \eta \, \frac{\partial \mathcal{E}}{\partial w_{c_j^v i}}, \quad w_{c_j^{in} i} -= \eta \, \frac{\partial \mathcal{E}}{\partial w_{c_j^{in} i}}, \quad \ldots$$



Kokkuvõte

- rekurrentsed neurovõrgud (RNN)
- nende õpetamine
  - epohhipõhiselt/pidevalt
  - BPTT/RTRL
- RNN lühike mälu
- pikk lühiajaline mälu (LSTM)



Aitäh tähelepanu eest!