Pokročilé metódy analýzy dát 4

úvod do hlbokého učenia

Peter Bednár

Priestorové dáta – rozpoznávanie obrazu

Rozpoznávanie obrazu

- Úlohou skrytých vrstiev pri rozpoznávaní obrazu je vyextrahovať vektor vizuálnych príznakov, na základe ktorých sa obraz klasifikuje do tried
- Farebný obrázok o rozmeroch 200x200 pixelov vyžaduje pri úplnom prepojení medzi vstupnou a skrytou vrstvou 200x200x3 = 120 000 váh (pre jeden neurón na skrytej vrstve) – počet veľmi rýchlo narastá s ďalšími vrstvami a pri väčšom rozlíšení obrazu

Konvolučná vrstva (1)

- Základný predpoklad:
  - Príznaky je možné rozpoznať v lokálnej ohraničenej oblasti
  - Ten istý príznak môže byť rozpoznaný v rôznych častiach vstupného obrazu
- Pre neurón na konvolučnej vrstve platí:
  - Váhy sú ohraničené iba na malú vstupnú oblasť (napr. 5x5 bodov)
  - Jeho aktivácia je vypočítaná postupne na celom vstupnom obrázku posúvaním oblasti v horizontálnom aj vertikálnom smere – matematicky ide o operáciu konvolúcie
  - Aktivácia je daná skalárnym súčinom váh a hodnôt vstupnej oblasti (+ voliteľne *bias*)

Konvolučná vrstva (2)

- Základné parametre:
  - Veľkosť vstupného "okna" (*size*) v horizontálnom aj vertikálnom smere (napr. 5x5 bodov)
  - Krok posunutia (*stride*) v horizontálnom aj vertikálnom smere (napr. po 1 bode v oboch smeroch)
  - Ak chceme aby bol zachovaný rozmer vstupných dát po vypočítaní konvolúcie, rozšírime obrázok o okraj v horizontálnom aj vertikálnom smere doplnením 0 hodnôt (*zero-padding*)

Neurón konvolučnej vrstvy (1)

Veľkosť vstupu: 5x5
Veľkosť okna: 3x3
Posunutie: 1,1
Okraj: 1,1,1,1

$w_0 = 1$

1	1	0
0	0	1
1	-1	0

0	0	0	0	0	0	0
0	2	2	2	0	1	0
0	1	2	0	1	2	0
0	0	1	0	2	0	0
0	1	1	2	0	2	0
0	1	2	0	1	0	0
0	0	0	0	0	0	0

Veľkosť výstupu: 5x5

1.0 + 1.0 + 0.0 +
0.0 + 0.2 + 1.2 +
1.0 + -1.1 + 0.2 + 1 = 2

2				

Neurón konvolučnej vrstvy (2)

Veľkosť vstupu: 5x5
Veľkosť okna: 3x3
Posunutie: 1,1
Okraj: 1,1,1,1

$w_0 = 1$

1	1	0
0	0	1
1	-1	0

0	0	0	0	0	0	0
0	2	2	2	0	1	0
0	1	2	0	1	2	0
0	0	1	0	2	0	0
0	1	1	2	0	2	0
0	1	2	0	1	0	0
0	0	0	0	0	0	0

Veľkosť výstupu: 5x5

1.0 + 1.0 + 0.0 +
0.2 + 0.2 + 1.2 +
1.1 + -1.2 + 0.0 + 1 = 2

2	2			

Neurón konvolučnej vrstvy (3)

Veľkosť vstupu: 5x5
Veľkosť okna: 3x3
Posunutie: 1,1
Okraj: 1,1,1,1

$w_0 = 1$

1	1	0
0	0	1
1	-1	0

0	0	0	0	0	0	0
0	2	2	2	0	1	0
0	1	2	0	1	2	0
0	0	1	0	2	0	0
0	1	1	2	0	2	0
0	1	2	0	1	0	0
0	0	0	0	0	0	0

Veľkosť výstupu: 5x5

1.0 + 1.0 + 0.0 +
0.2 + 0.2 + 1.0 +
1.2 + -1.0 + 0.1 + 1 = 3

2	2	3		

Neurón konvolučnej vrstvy (4)

Veľkosť vstupu: 5x5
Veľkosť okna: 3x3
Posunutie: 1,1
Okraj: 1,1,1,1

$w_0 = 1$

1	1	0
0	0	1
1	-1	0

0	0	0	0	0	0	0
0	2	2	2	0	1	0
0	1	2	0	1	2	0
0	0	1	0	2	0	0
0	1	1	2	0	2	0
0	1	2	0	1	0	0
0	0	0	0	0	0	0

Veľkosť výstupu: 5x5

1.0 + 1.0 + 0.0 +
0.2 + 0.0 + 1.1 +
1.0 + -1.1 + 0.2 + 1 = 1

2	2	3	1	

Neurón konvolučnej vrstvy (5)

Veľkosť vstupu: 5x5
Veľkosť okna: 3x3
Posunutie: 1,1
Okraj: 1,1,1,1

$w_0 = 1$

0	0	0	0	0	0	0
0	2	2	2	0	1	0
0	1	2	0	1	2	0
0	0	1	0	2	0	0
0	1	1	2	0	2	0
0	1	2	0	1	0	0
0	0	0	0	0	0	0

1	1	0
0	0	1
1	-1	0

Veľkosť výstupu: 5x5

1.0 + 1.0 + 0.0 +
0.0 + 0.1 + 1.0 +
1.1 + -1.2 + 0.1 + 1 = 0

2	2	3	1	0

Neurón konvolučnej vrstvy (6)

Veľkosť vstupu: 5x5
Veľkosť okna: 3x3
Posunutie: 1,1
Okraj: 1,1,1,1

$w_0 = 1$

1	1	0
0	0	1
1	-1	0

0	0	0	0	0	0	0
0	2	2	2	0	1	0
0	1	2	0	1	2	0
0	0	1	0	2	0	0
0	1	1	2	0	2	0
0	1	2	0	1	0	0
0	0	0	0	0	0	0

Veľkosť výstupu: 5x5

1.0 + 1.2 + 0.2 +
0.0 + 0.1 + 1.2 +
1.0 + -1.0 + 0.1 + 1 = 5

2	2	3	1	0
5				

Neurón konvolučnej vrstvy (7)

Veľkosť vstupu: 5x5
Veľkosť okna: 3x3
Posunutie: 1,1
Okraj: 1,1,1,1

$w_0 = 1$

1	1	0
0	0	1
1	-1	0

0	0	0	0	0	0	0
0	2	2	2	0	1	0
0	1	2	0	1	2	0
0	0	1	0	2	0	0
0	1	1	2	0	2	0
0	1	2	0	1	0	0
0	0	0	0	0	0	0

Veľkosť výstupu: 5x5

2	2	3	1	0
5	4	7	3	4
2	4	4	4	2
1	3	4	4	4
4	3	5	3	3

Konvolučná vrstva (3)

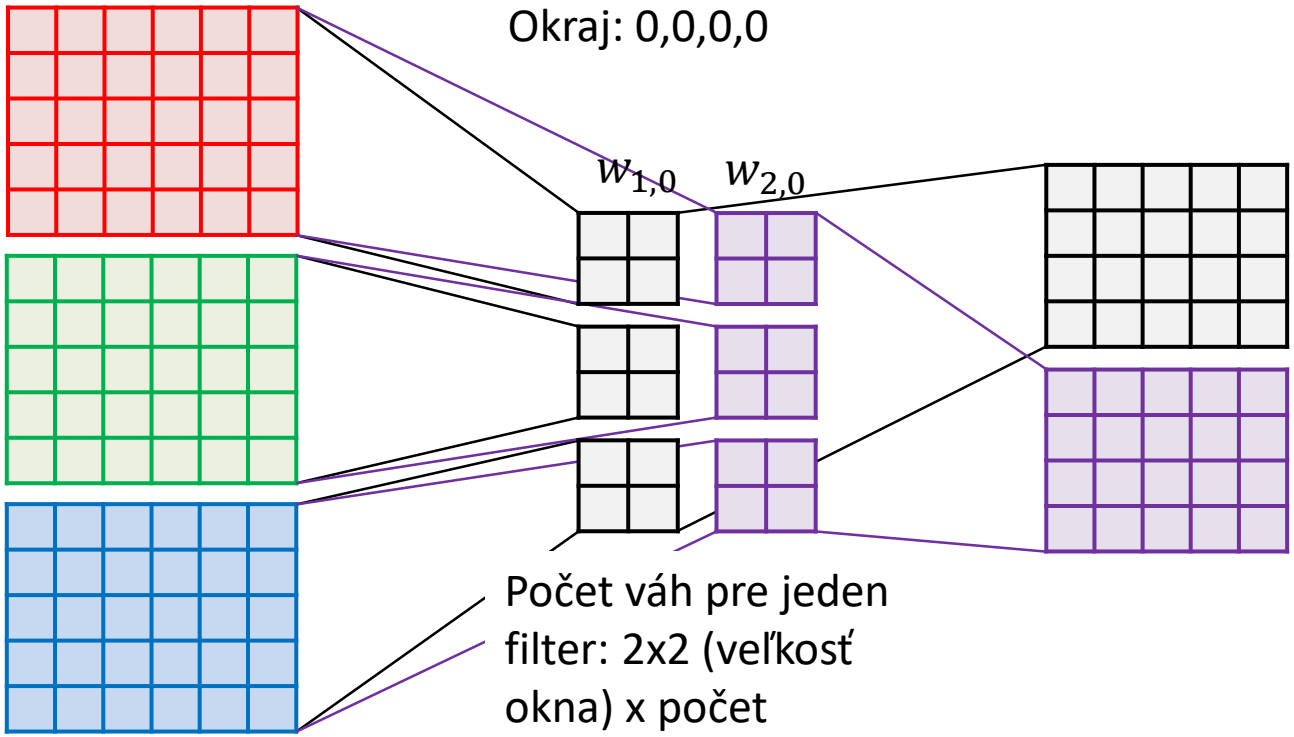
- Na konvolučnej vrstve môžeme mať viac neurónov – filtrov.
  - Veľkosť výstupu je daný (veľkosťou vstupu/okna, posunom, okrajom) x počet filtrov
- Na vstupe môžeme mať viac atribútov (kanálov) pre jeden bod :
  - Štandardne pre farebný obrázok R G B zložky, ale napr. pre satelitné obrázky môžeme mať stovky kanálov pre rôzne oblasti elektromagnetického spektra
  - Počet váh pre jeden neurón: šírka okna x výška okna x počet kanálov (+ voliteľne jeden *bias*)

Konvolučná vrstva – príklad

Veľkosť vstupu:
6x5xpočet vstupných kanálov

2 filtre
Veľkosť okna: 2x2
Posunutie: 1,1
Okraj: 0,0,0,0

Veľkosť výstupu: 5x4x počet filtrov (neurónov na konvolučnej vrstve)



Počet váh pre jeden filter: 2x2 (veľkosť okna) x počet vstupných kanálov + 1

Konvolučné siete

- Váhy filtrov sú na začiatku inicializované náhodne, ale postupne sa filtre naučia rozoznávať užitočné vzory vo vstupných dátach (napr. hrany, farebné prechody a pod.)
- Zreťazením viacerých konvolučných vrstiev je možné vytvoriť hierarchiu viac a viac zložitejších vizuálnych vzorov.
- Pri učení sa gradient pre konvolučné neuróny počíta tiež konvolúciou, ale v opačnom smere ako pri výpočte predikcie
- Aby sme mohli na výstupe obraz efektívne klasifikovať úplne prepojenou doprednou sieťou, je potrebné postupne redukovať výstupné rozmery konvolučných vrstiev

Redukcia výstupných rozmerov

- Môžeme použiť väčší krok, napr. ak budeme posúvať okno 2x2 s krokom 2 vo vertikálnom aj horizontálnom smere, vstupný obrázok 10x10 môžeme zredukovať na 5x5
- Môžeme použiť vrstvu, ktorá zlúči oblasti a zagreguje hodnoty pre celé okno - tzv. *pooling*
  - Ako agregačná funkcia sa najčastejšie používa maximum, ale je možné použiť aj ďalšie funkcie (napr. priemer, alebo Euklidovskú normu)
  - Pooling zachováva počet vstupných kanálov

Redukcia výstupných rozmerov - príklad

- Max-pooling s veľkosťou okna 2x2 a posunom 2,2 (pri poolingu sa väčšinou oblasti neprekrývajú):

Veľkosť vstupu: 6x6

1	2	0	1	0	1
0	3	0	0	0	-2
1	2	0	1	0	1
2	-3	0	1	1	0
1	4	1	1	0	-1
2	3	2	2	0	-1

Veľkosť výstupu: 3x3
(jeden bod pre celé okno 2x2)

3	1	1
2	1	1
4	2	0

Príklad architektúry - VGGNet (1)

- Vstup: obrázok 224x224x3
- Výstup: klasifikácia do 1000 tried (objektov)
- Homogénna architektúra zložená z konvolučných vrstiev s oknom 3x3 a posunom 1x1 (okraj 1) a max poolingu 2x2
- Celkovo 13 konvolučných vrstiev CONV3 + ReLU transformácia + 5 pooling vrstiev POOL2 + 3 FC vrstvy doprednej siete pre výslednú klasifikáciu – 138M váh (ale z toho až 100M na prvej FC vrstve)

Príklad architektúry - VGGNet (2)

	Vrstva	Rozmery		Vrstva	Rozmery
1	INPUT	224x224x3	12	CONV3-512	28x28x512
2	CONV3-64	224x224x64	13	CONV3-512	28x28x512
3	CONV3-64	224x224x64	14	CONV3-512	28x28x512
4	POOL2	112x112x64	15	POOL2	14x14x512
5	CONV3-128	112x112x128	16	CONV3-512	14x14x512
6	CONV3-128	112x112x128	17	CONV3-512	14x14x512
7	POOL2	56x56x128	18	CONV3-512	14x14x512
8	CONV3-256	56x56x256	19	POOL2	7x7x512
9	CONV3-256	56x56x256	20	FC	4096
10	CONV3-256	56x56x256	21	FC	4096
11	POOL2	28x28x256	22	FC	1000

Vizualizácia konvolučných sietí

https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

Sekvencie

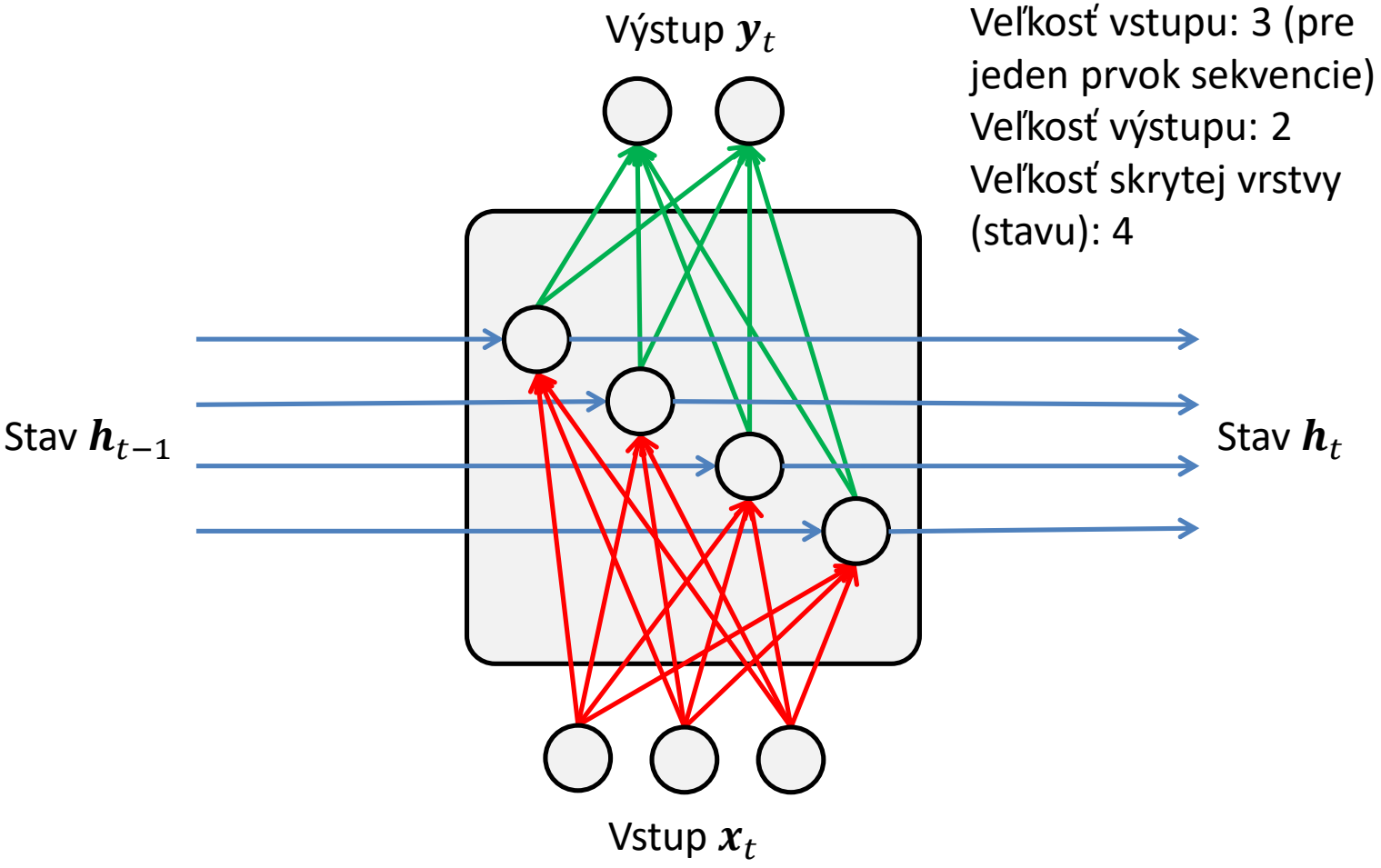
Prediktívne úlohy učenia so sekvenciami

- Klasifikácia sekvencií: sekvencia → trieda
  - Napr. analýza sentimentu vo vete – pozitívna/negatívna/neutrálna
- Mapovanie sekvencií: sekvencia → sekvencia
  - Rovnako dlhé sekvencie: napr. extrahovanie názvov z textov (o každom slove chceme rozhodnúť či je súčasťou názvu, alebo nie)
  - Rôzne dlhé sekvencie: napr. rozpoznávanie reči (vstup je časová postupnosť úrovní audio signálu, výstup textový prepis, resp. sekvencia hlások)

Rekurentné siete (1)

- Základný model pre mapovanie sekvencií rovnakej dĺžky.
- Vstup: sekvencia $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T$
- Výstup: sekvencia $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T$
  - Každý prvok sekvencie $\boldsymbol{x}_t$, alebo $\boldsymbol{y}_t$, môže byť číselný vektor
- Skrytá vrstva udržiava stav $\boldsymbol{h}_t$ (vektor váh), ktorý je závislý od vstupu $\boldsymbol{x}_t$ a predchádzajúceho stavu $\boldsymbol{h}_{t-1}$ (ktorý je rekurzívne závislý od všetkých predchádzajúcich vstupov) a z ktorého je odvodený výstup $\boldsymbol{y}_t$
- Počiatočný stav $\boldsymbol{h}_0$ je inicializovaný na $\mathbf{0}$

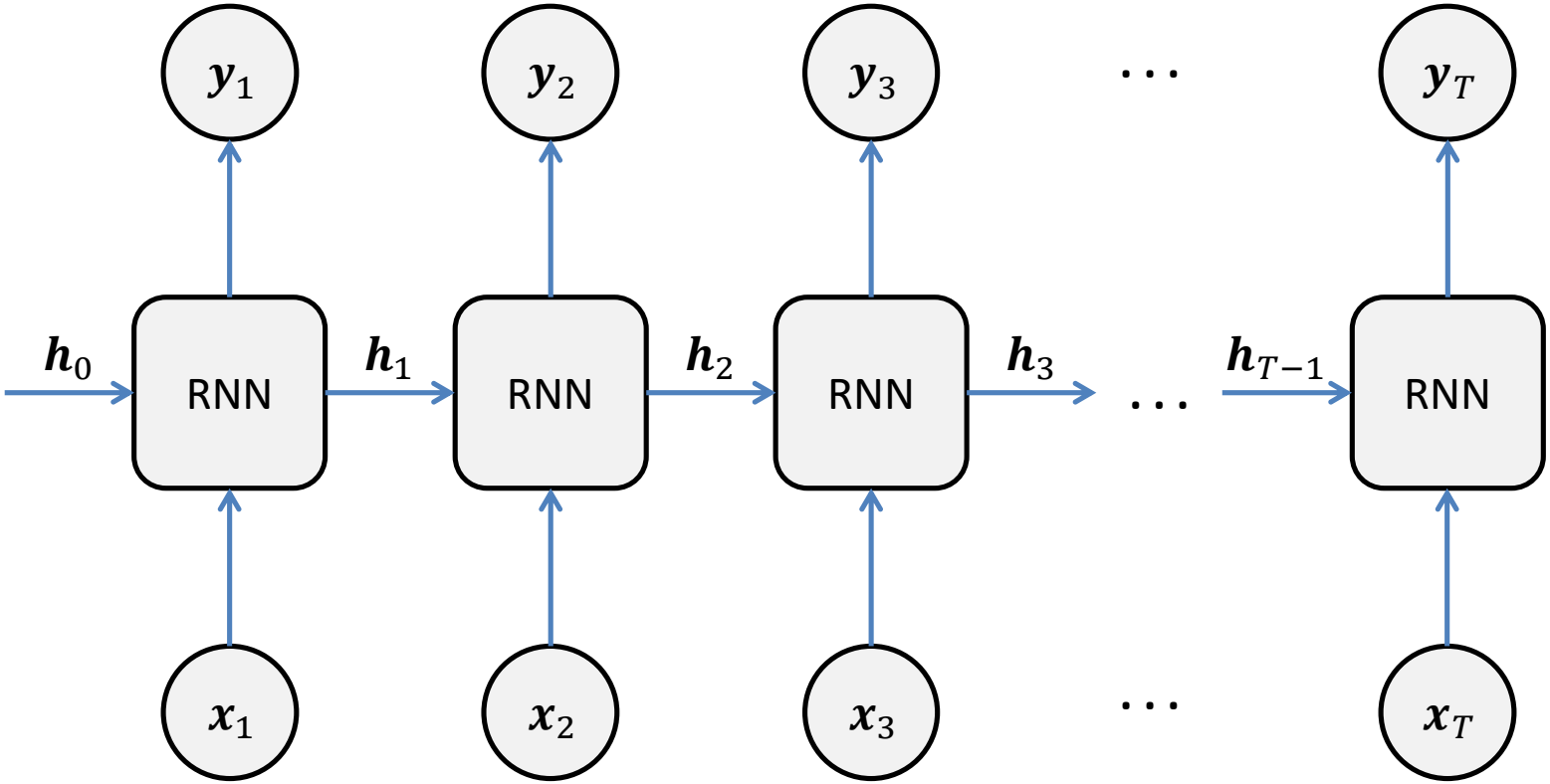
Rekurentné siete (2)



Spätné šírenie chyby v čase (1)

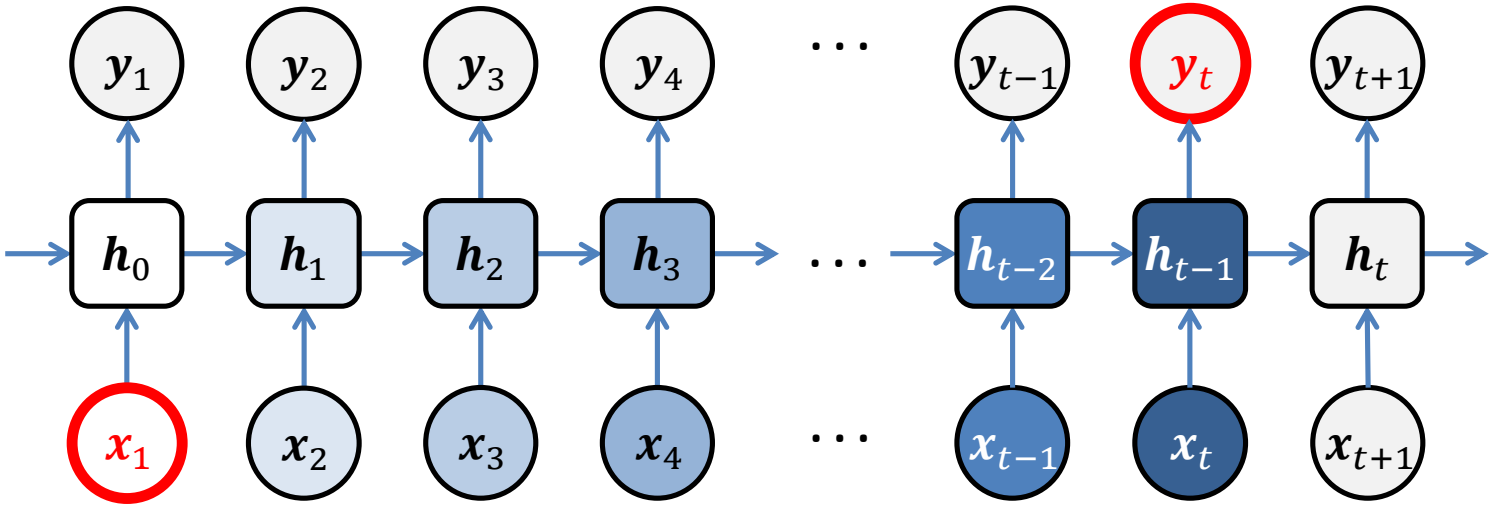
- Pre jeden príklad (vstupnú sekvenciu $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T$ a výstupnú sekvenciu $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T$)
  - Pre každý vstupný prvok $\boldsymbol{x}_t$ postupne vypočítame výstupnú predikciu a nasledujúci stav $\boldsymbol{h}_t$ skombinovaním predchádzajúceho stavu $\boldsymbol{h}_{t-1}$ a vstupu $\boldsymbol{x}_t$
  - Vypočítame chybu predikcie pre celú výstupnú sekvenciu porovnaním s $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T$
  - Rekurzívne prepočítame váhy štandardnou gradientovou metódou
- Graficky môžeme učenie "rozbaliť v čase" do nasledujúcej siete, kde každý blok zdieľa tie isté váhy

Spätné šírenie chyby v čase (2)



Rekurentné siete (3)

- Pre dlhé sekvencie sa rekurentná sieť rozbalí do veľmi hlbokej siete – viac náchylné na vymiznutie gradientu
- Problematické je hlavne učenie vzdialených závislostí medzi vstupom a výstupom



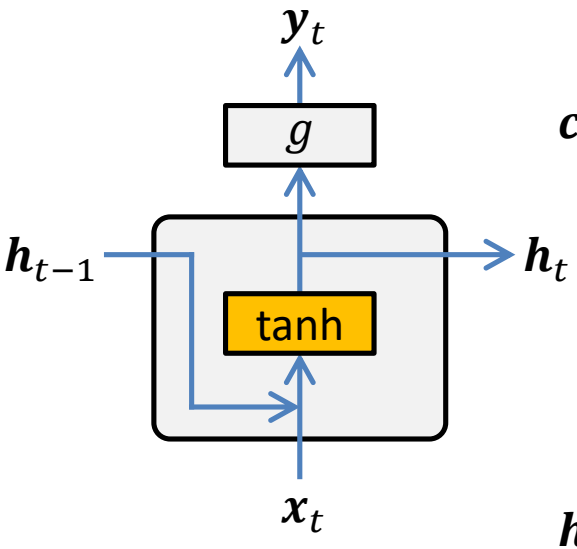
Long-Short Term Memory (1)

- Architektúra špeciálne navrhnutá pre učenie vzdialených závislostí
- Okrem stavu $\boldsymbol{h}_t$ udržiava aj pamäť $\boldsymbol{c}_t$ (vektor váh) v ktorej je možné priamo uchovať informácie zo vzdialených predchádzajúcich stavov
- Pre riadenie "zapisovania/mazania" pamäte sa využívajú hradlá: neuróny s logistickou aktivačnou funkciou
  - pamäť × 0 – "mazanie"
  - hodnota × 1 + pamäť – "zapisovanie"

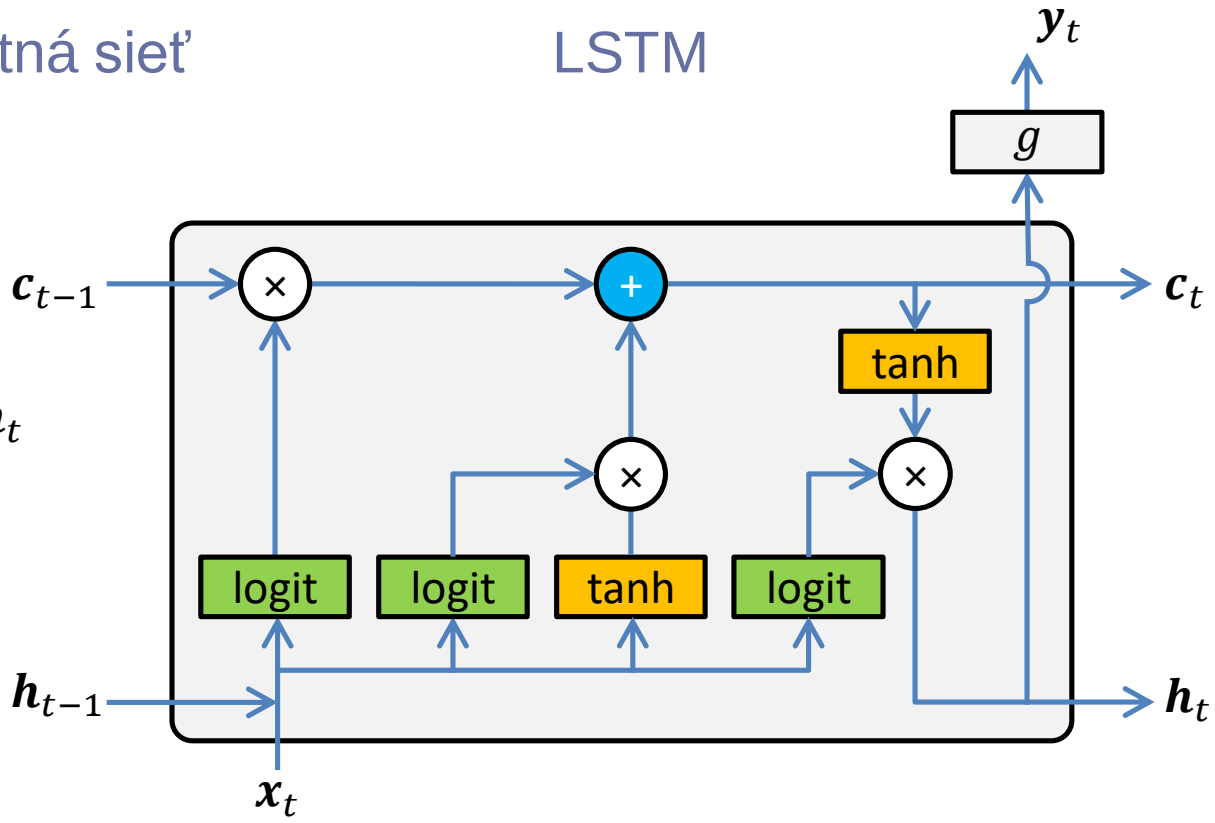
Long-Short Term Memory (2)

- Pre vysvetlenie zavedieme blokovú schému.

Štandardná rekurentná sieť

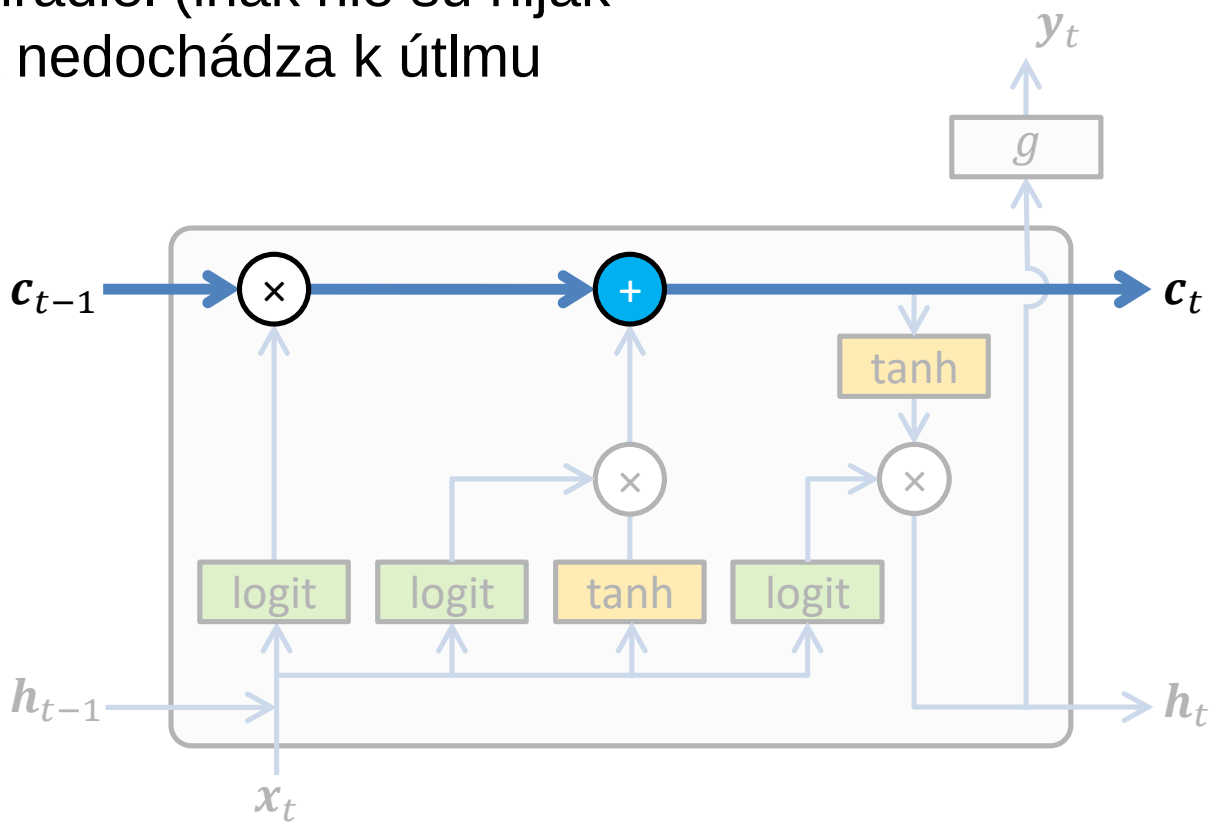


LSTM



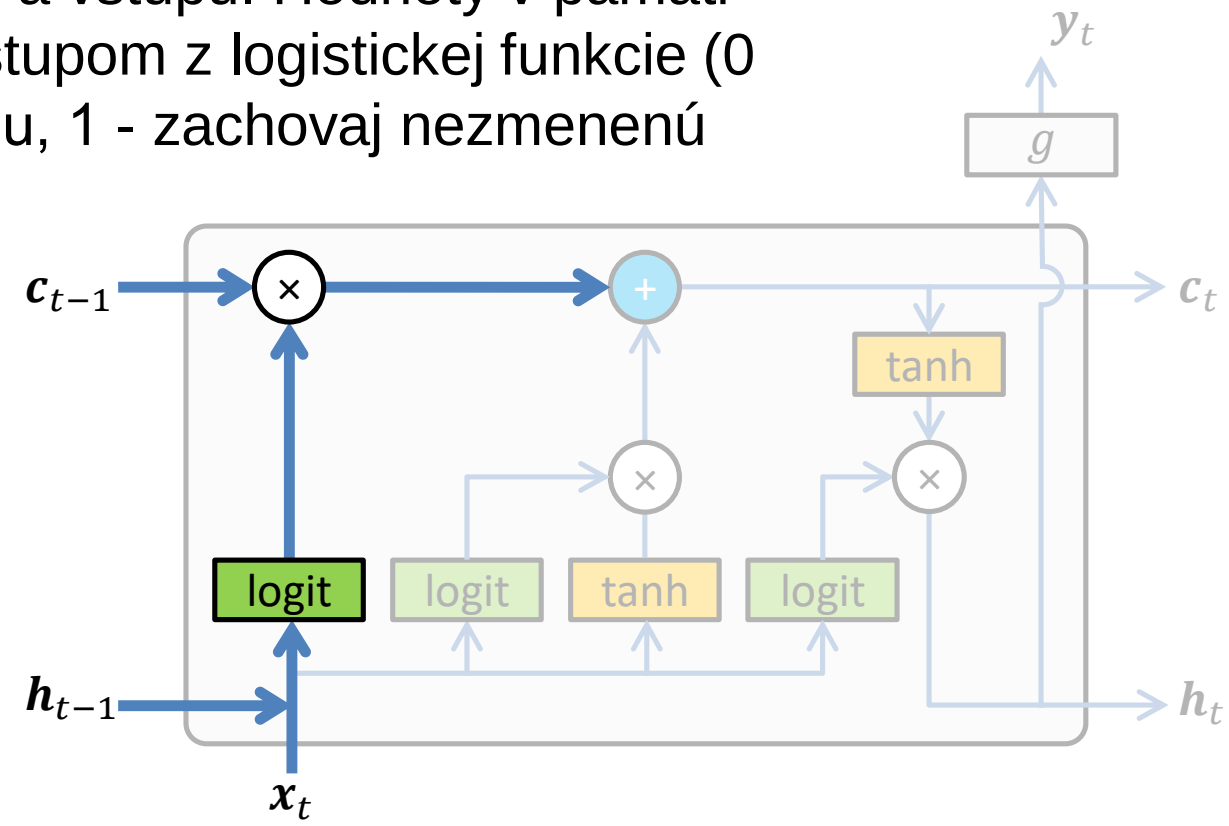
Long-Short Term Memory (3)

- Informácie v pamäti sú modifikované iba prostredníctvom hradiel (inak nie sú nijak transformované a nedochádza k útlmu informácií)



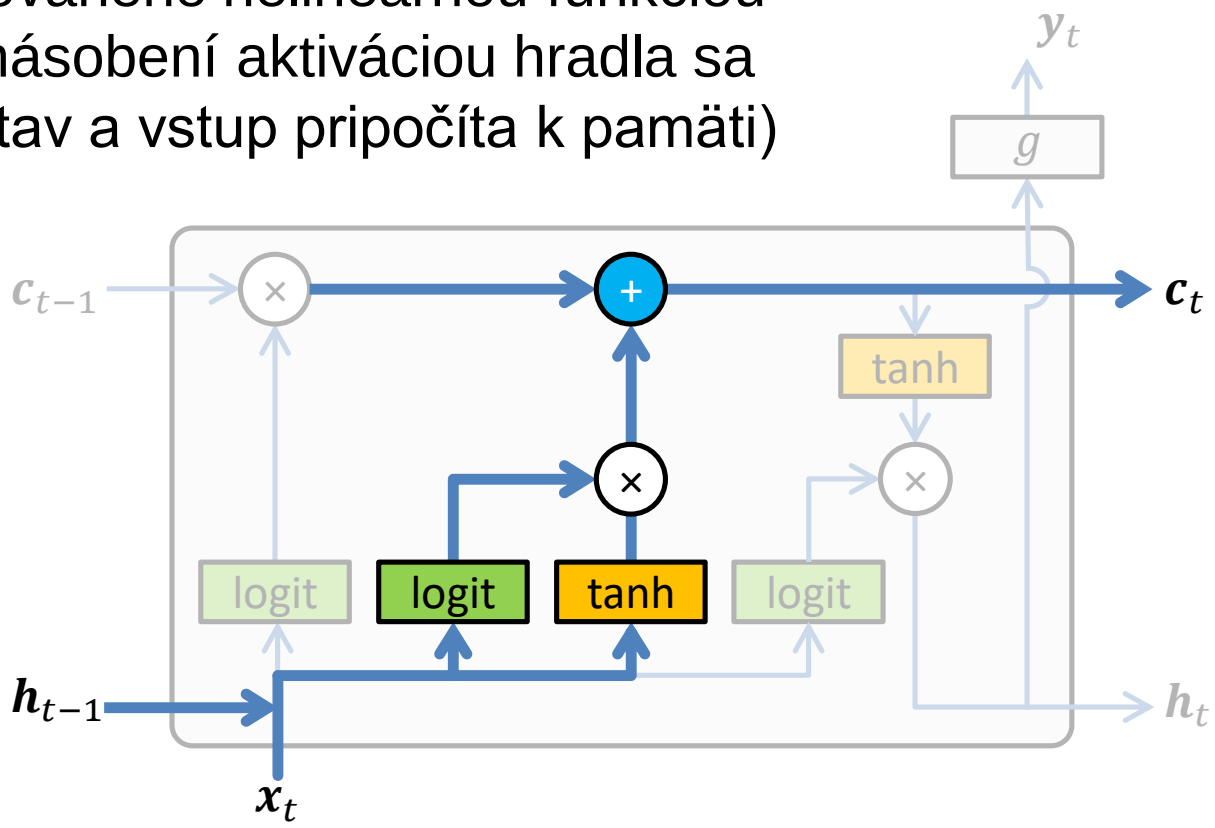
Long-Short Term Memory (4)

- Prvé hradlo riadi mazanie z pamäti podľa aktuálneho stavu a vstupu. Hodnoty v pamäti sa vynásobia výstupom z logistickej funkcie (0 – zmaž informáciu, 1 - zachovaj nezmenenú informáciu)



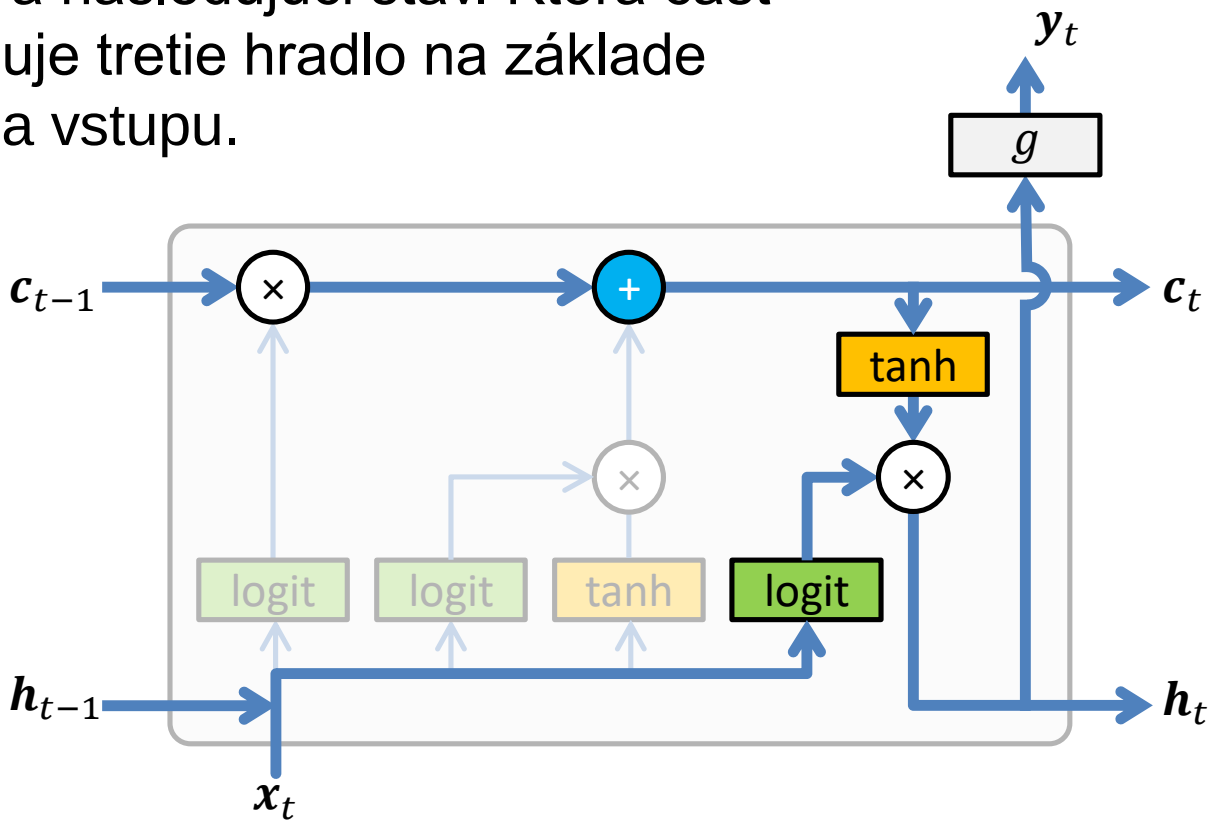
Long-Short Term Memory (5)

- Druhé hradlo riadi zápis aktuálneho stavu a vstupu transformovaného nelineárnou funkciou do pamäti (po vynásobení aktiváciou hradla sa transformovaný stav a vstup pripočíta k pamäti)



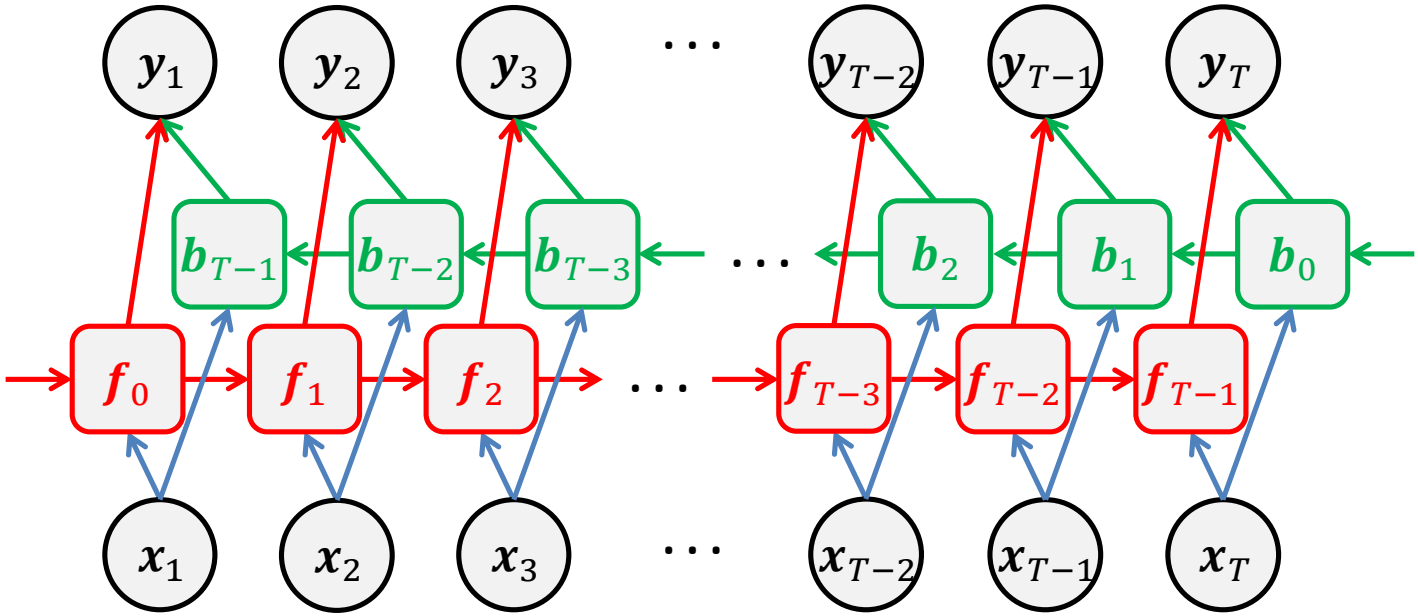
Long-Short Term Memory (6)

- Časť z pamäti sa po nelineárnej transformácii použije na výstup a nasledujúci stav. Ktorá časť sa použije rozhoduje tretie hradlo na základe aktuálneho stavu a vstupu.



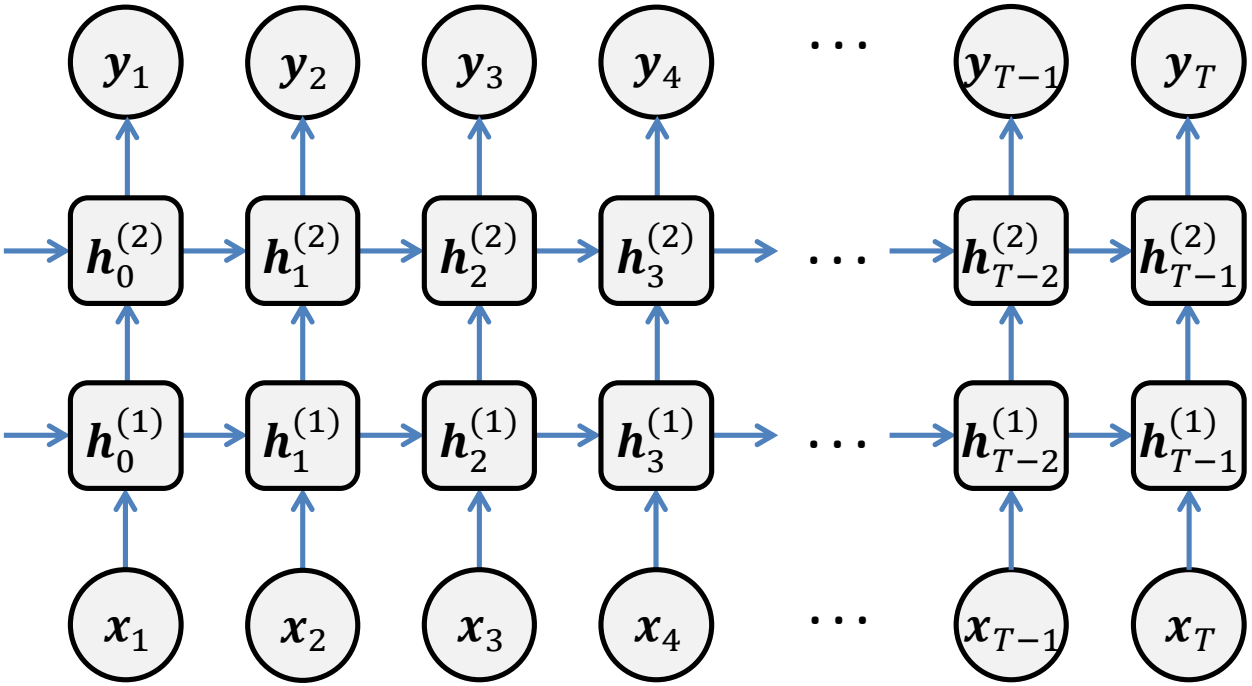
Obojsmerné rekurentné siete

- Pri niektorých aplikáciách má zmysel pri výpočte výstupu uvažovať nielen predchádzajúci kontext, ale aj nasledujúci. Môžeme skombinovať dve rekurentné siete, ktoré prechádzajú sekvenciu v oboch smeroch



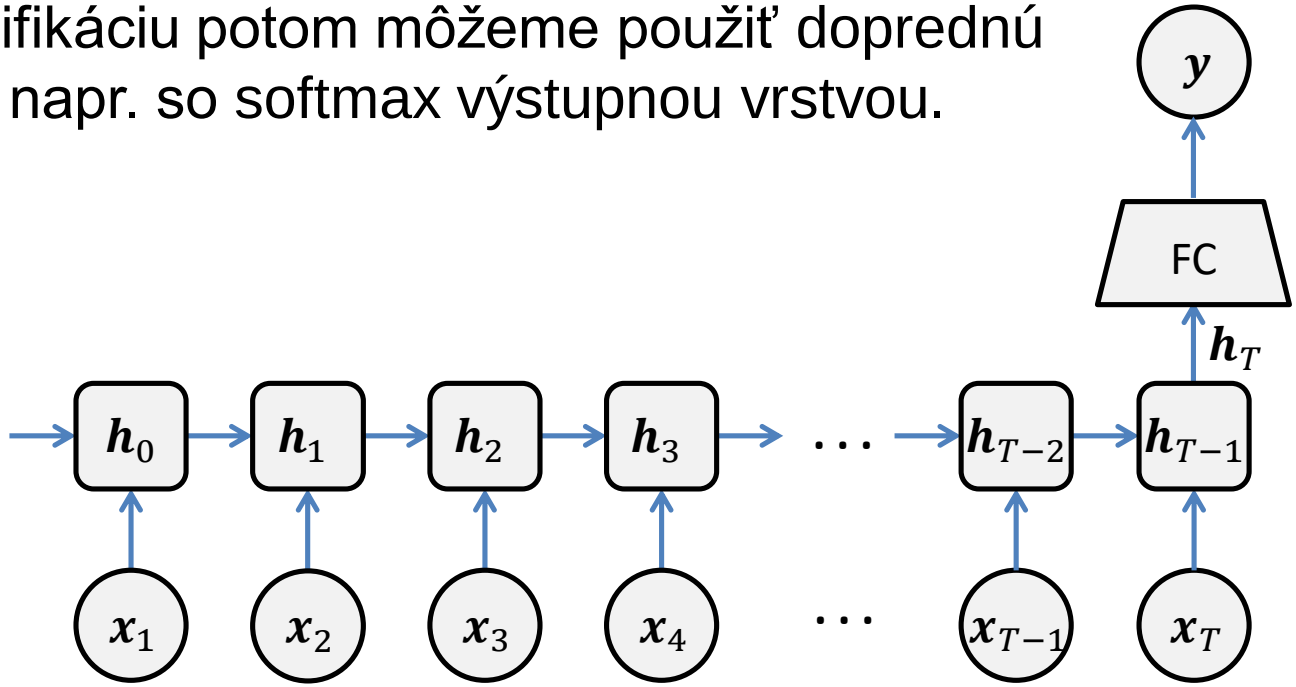
Hlboké rekurentné siete

- Pre zložitejšie závislosti môžeme navrstviť viacero rekurentných sietí (stav nižšej vrstvy sa použije ako vstup vyššej). Podobne môžeme skombinovať aj obojsmerné siete.



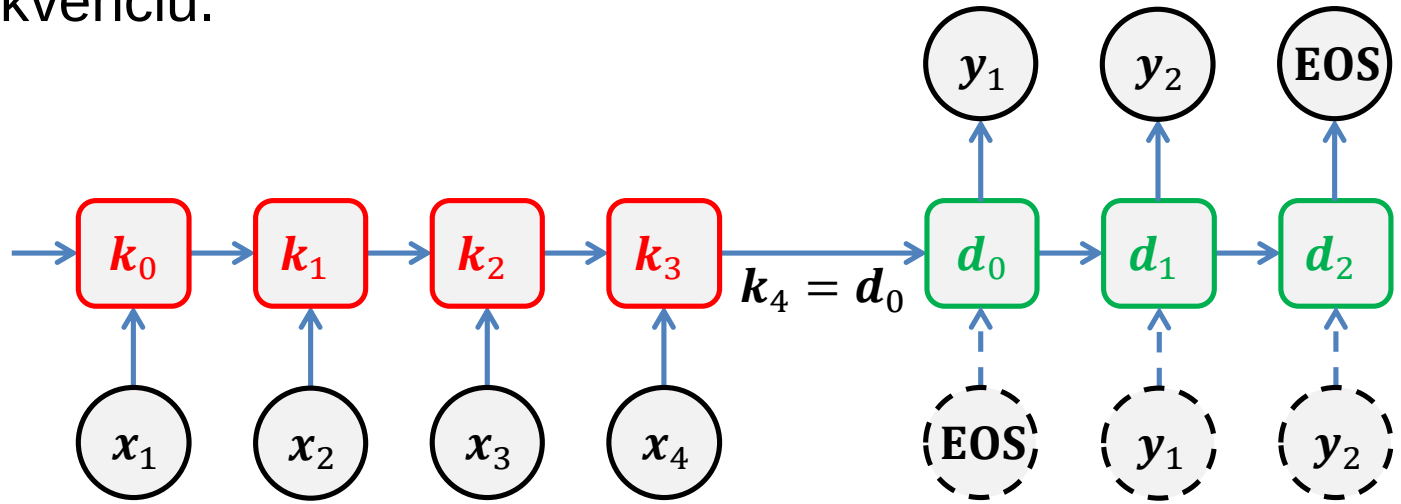
Klasifikácia sekvencií

- Posledný stav siete predstavuje vektorovú reprezentáciu celej vstupnej sekvencie. Na klasifikáciu potom môžeme použiť doprednú sieť napr. so softmax výstupnou vrstvou.



Mapovanie sekvencií rôznych dĺžok

- Používa sa kodér/dekodér architektúra. Kodér zakóduje vstupnú sekvenciu ako číselný vektor, ktorý sa použije ako počiatočný stav pre dekodér, ktorý vygeneruje výstupnú sekvenciu.



EOS – špeciálny vstup/výstup, ktorý reprezentuje začiatok/koniec výstupnej sekvencie