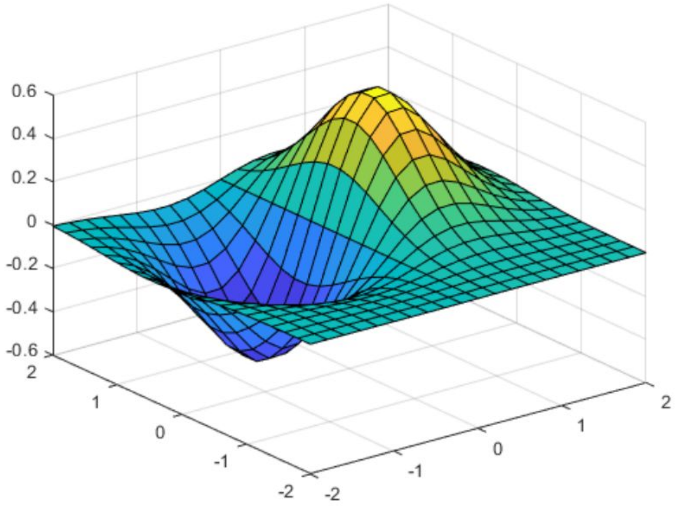
Jak se neuronové sítě učí

Backpropagation



Backpropagation

- Hadamardův produkt:

Backpropagation

- Hadamardův produkt:

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \odot \begin{pmatrix} 2 \\ 4 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \cdot 2 \\ 2 \cdot 4 \\ 3 \cdot 3 \end{pmatrix} = \begin{pmatrix} 2 \\ 8 \\ 9 \end{pmatrix}$$

Backpropagation - chyba neuronu

- připomenutí: chceme popsat, jak změna vah a biasů ovlivní účelovou funkci

Backpropagation - chyba neuronu

- připomenutí: chceme popsat, jak změna vah a biasů ovlivní účelovou funkci
- To znamená spočítat $\dfrac{\partial C}{\partial w^l_{jk}}$ a $\dfrac{\partial C}{\partial b^l_j}$

Backpropagation - chyba neuronu

- připomenutí: chceme popsat, jak změna vah a biasů ovlivní účelovou funkci
- To znamená spočítat $\frac{\partial C}{\partial w^l_{jk}}$ a $\frac{\partial C}{\partial b^l_j}$
- přitom w^l_{jk} je váha spojení mezi ***k***-tým neuronem ***(l-1)***-té vrstvy a j-tým neuronem ***l***-té vrstvy (indexy naopak) a b^l_j je bias ***j***-tého neuronu ***l***-té vrstvy

Backpropagation - chyba neuronu

- Zavedeme ***chybu j-tého neuronu v l-té vrstvě:*** δ_j^l

Backpropagation - chyba neuronu

- Zavedeme ***chybu j-tého neuronu v l-té vrstvě:*** δ_j^l
- Trošku změníme chování neuronu, takže místo

  $\sigma\left(z_j^l\right)$ je výstup z neuronu $\sigma\left(z_j^l + \Delta z_j^l\right)$

Backpropagation - chyba neuronu

- Zavedeme ***chybu j-tého neuronu v l-té vrstvě:*** δ_j^l
- Trošku změníme chování neuronu, takže místo

$\sigma\left(z_j^l\right)$ je výstup z neuronu $\sigma\left(z_j^l + \Delta z_j^l\right)$

- To vede ke změně účelové funkce o $\frac{\partial C}{\partial z_j^l} \Delta z_j^l$

Backpropagation - chyba neuronu

- Zavedeme ***chybu j-tého neuronu v l-té vrstvě:*** δ^l_j
- Trošku změníme chování neuronu, takže místo $\sigma\left(z^l_j\right)$ je výstup z neuronu $\sigma\left(z^l_j + \Delta z^l_j\right)$
- To vede ke změně účelové funkce o $\frac{\partial C}{\partial z^l_j}\Delta z^l_j$
- Pokud je $\frac{\partial C}{\partial z^l_j}$ velké, můžeme hodnotu úč. funkce snížit volbou Δz^l_j (s opačným znaménkem)

Backpropagation - chyba neuronu

- Zavedeme ***chybu j-tého neuronu v l-té vrstvě:*** δ_j^l
- Trošku změníme chování neuronu, takže místo
  $\sigma(z_j^l)$ je výstup z neuronu $\sigma(z_j^l + \Delta z_j^l)$
- To vede ke změně účelové funkce o $\frac{\partial C}{\partial z_j^l} \Delta z_j^l$

- Pokud je $\frac{\partial C}{\partial z_j^l}$ velké, můžeme hodnotu úč. funkce snížit volbou Δz_j^l (s opačným znaménkem)
- Pokud je malé, nic moc nezmůžeme.

Backpropagation - chyba neuronu

- $\frac{\partial C}{\partial z_j^l}$ se tedy dá chápat jako chyba konkrétního neuronu

Backpropagation - chyba neuronu

- $\frac{\partial C}{\partial z_j^l}$ se tedy dá chápat jako chyba konkrétního neuronu
- Proto ***chyba j-tého neuronu v l-té vrstvě***

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l}$$

Backpropagation - chyba neuronu

- $\frac{\partial C}{\partial z_j^l}$ se tedy dá chápat jako chyba konkrétního neuronu
- Proto ***chyba j-tého neuronu v l-té vrstvě***

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l}$$

- Vektor chyb ***l***-té vrstvy označíme jako δ^l

Backpropagation - chyba neuronu

- $\frac{\partial C}{\partial z_j^l}$ se tedy dá chápat jako chyba konkrétního neuronu
- Proto ***chyba j-tého neuronu v l-té vrstvě***

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l}$$

- Vektor chyb ***l***-té vrstvy označíme jako δ^l
- Proč neměníme přímo output neuronu (tj. po aplikaci aktivační funkce)?

Backpropagation - chyba neuronu

- $\frac{\partial C}{\partial z_j^l}$ se tedy dá chápat jako chyba konkrétního neuronu
- Proto ***chyba j-tého neuronu v l-té vrstvě***

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l}$$

- Vektor chyb ***l***-té vrstvy označíme jako δ^l
- Proč neměníme přímo output neuronu (tj. po aplikaci aktivační funkce)? **Odvození je pak pracnější, ale mohli bychom**

Backpropagation - 4 rovnice algoritmu

1. Chyba výstupní vrstvy δ^L se rovná: $\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L)$

Backpropagation - 4 rovnice algoritmu

1. Chyba výstupní vrstvy δ^L se rovná: $\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L)$

- $\frac{\partial C}{\partial a_j^L}$ popisuje, jak se změní **C** v závislosti na ***j***-té aktivaci (pokud na tomto neuronu příliš nezáleží, bude malé)

Backpropagation - 4 rovnice algoritmu

1. Chyba výstupní vrstvy δ^L se rovná: $\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L)$

- $\frac{\partial C}{\partial a_j^L}$ popisuje, jak se změní **C** v závislosti na ***j***-té aktivaci (pokud na tomto neuronu příliš nezáleží, bude malé)
- $\sigma'(z_j^L)$ popisuje, jak se mění aktivace (výstup neuronu) na základě z_j^L . Tedy jak závisí aktivační funkce na změně z_j^L

Backpropagation - 4 rovnice algoritmu

1. Chyba výstupní vrstvy δ^L se rovná: $\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L)$

Backpropagation - 4 rovnice algoritmu

1. Chyba výstupní vrstvy δ^L se rovná: $\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L)$

- Přepíšeme jako: $\delta^L = \begin{pmatrix} \frac{\partial C}{\partial a_1^L} \\ \vdots \\ \frac{\partial C}{\partial a_n^L} \end{pmatrix} \odot \begin{pmatrix} \sigma'(z_1^L) \\ \vdots \\ \sigma'(z_n^L) \end{pmatrix} = \nabla_a C \odot \sigma'(z^L)$

Backpropagation - 4 rovnice algoritmu

1. Chyba výstupní vrstvy δ^L se rovná: $\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L)$

- Přepíšeme jako: $\delta^L = \begin{pmatrix} \frac{\partial C}{\partial a_1^L} \\ \vdots \\ \frac{\partial C}{\partial a_n^L} \end{pmatrix} \odot \begin{pmatrix} \sigma'(z_1^L) \\ \vdots \\ \sigma'(z_n^L) \end{pmatrix} = \nabla_a C \odot \sigma'(z^L)$

- Pro MSE:

$$\nabla_a C = (a^L - y) \implies \delta^L = (a^L - y) \odot \sigma'(z^L)$$

Backpropagation - 4 rovnice algoritmu

2. Chyba *l*-té vrstvy v závislosti na *(l+1)* vrstvě:

$$\delta^l = \left(\left(w^{l+1}\right)^T \delta^{l+1}\right) \odot \sigma'\left(z^l\right)$$

Backpropagation - 4 rovnice algoritmu

2. Chyba *l*-té vrstvy v závislosti na *(l+1)* vrstvě:

$$\delta^l = \left(\left(w^{l+1} \right)^T \delta^{l+1} \right) \odot \sigma' \left(z^l \right)$$

- když známe chybu ***(l+1)*** vrstvy, vynásobíme ji transponovanou maticí vah (jako bychom chtěli tuto chybu posunout do předchozí vrstvy)

Backpropagation - 4 rovnice algoritmu

2. Chyba *l*-té vrstvy v závislosti na *(l+1)* vrstvě:

$$\delta^l = \left(\left(w^{l+1} \right)^T \delta^{l+1} \right) \odot \sigma' \left(z^l \right)$$

- když známe chybu ***(l+1)*** vrstvy, vynásobíme ji transponovanou maticí vah (jako bychom chtěli tuto chybu posunout do předchozí vrstvy)
- nyní dokážeme spočítat chybu jakékoliv vrstvy

Backpropagation - 4 rovnice algoritmu

3. Změna C na základě změny libovolného biasu:

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l$$

Backpropagation - 4 rovnice algoritmu

3. Změna C na základě změny libovolného biasu: $\frac{\partial C}{\partial b_j^l} = \delta_j^l$

- tedy chyba neuronu se rovná poměru změny ***C*** ku změně příslušného biasu

Backpropagation - 4 rovnice algoritmu

3. Změna C na základě změny libovolného biasu:

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l$$

- tedy chyba neuronu se rovná poměru změny ***C*** ku změně příslušného biasu
- tyto chyby už dokážeme spočítat díky předchozím dvěma rovnicím

Backpropagation - 4 rovnice algoritmu

4. Změna C na základě změny libovolné váhy:

$$\frac{\partial C}{\partial w_{jk}^{l}} = a_k^{l-1}\delta_j^l$$

Backpropagation - 4 rovnice algoritmu

4. Změna C na základě změny libovolné váhy:

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$$

- tuto parciální derivaci tedy můžeme spočítat pomocí chyby neuronu a aktivace předchozího neuronu (což už umíme)

Backpropagation - 4 rovnice algoritmu

4. Změna C na základě změny libovolné váhy:

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$$

- tuto parciální derivaci tedy můžeme spočítat pomocí chyby neuronu a aktivace předchozího neuronu (což už umíme)
- schematicky lze zapsat jako: $\frac{\partial C}{\partial w} \equiv a_{\rm in}\delta_{\rm out}$

Backpropagation - 4 rovnice algoritmu

4. Změna C na základě změny libovolné váhy:

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$$

- tuto parciální derivaci tedy můžeme spočítat pomocí chyby neuronu a aktivace předchozího neuronu (což už umíme)
- schematicky lze zapsat jako: $\frac{\partial C}{\partial w} \equiv a_{\rm in} \delta_{\rm out}$
- pokud je aktivace malá, parc. derivace je taky malá => prvek gradientu je malý => váha se učí pomalu

Backpropagation - 4 rovnice algoritmu

- váha se bude učit pomalu, pokud její input neuron má malou aktivaci (výstup je malý), nebo pokud má výstupní neuron nízkou/vysokou aktivaci

Backpropagation - 4 rovnice algoritmu

- váha se bude učit pomalu, pokud její input neuron má malou aktivaci (výstup je malý), nebo pokud má výstupní neuron nízkou/vysokou aktivaci
- rovnice budou platit pro obecnou aktivační funkci => to nám umožní definovat aktivační funkce s určitými učícími vlastnostmi (např. tak, abychom zabránili nízké aktivaci/saturaci neuronů a učení tak nebylo pomalé

Backpropagation - 4 rovnice algoritmu

- **Shrnutí:**

1. $\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'\left(z_j^L\right)$ zapíšeme jako $\nabla_a C \odot \sigma'\left(z^L\right)$

2. $\delta^l = \left(\left(w^{l+1}\right)^T \delta^{l+1}\right) \odot \sigma'\left(z^l\right)$, po prvcích $\delta_j^l = \sum_{k=1}^{n_{l+1}} w_{kj}^{l+1} \delta_k^{l+1} \sigma'\left(z_j^l\right)$

3. $\frac{\partial C}{\partial b_j^l} = \delta_j^l$

4. $\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$

Backpropagation - 4 rovnice algoritmu: důkaz

- **Důkaz 1. :** $\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'\left(z_j^L\right)$

Backpropagation - 4 rovnice algoritmu: důkaz

- **Důkaz 1. :** $\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'\left(z_j^L\right)$
- Vyjdeme z definice a aplikujeme derivaci složené funkce:

Backpropagation - 4 rovnice algoritmu: důkaz

- **Důkaz 1. :** $\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'\left(z_j^L\right)$
- Vyjdeme z definice a aplikujeme derivaci složené funkce:

$$\delta_j^L \equiv \frac{\partial C}{\partial z_j^L} = \sum_{k=1}^{n_L} \frac{\partial C}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L}$$

Backpropagation - 4 rovnice algoritmu: důkaz

- **Důkaz 1. :** $\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'\left(z_j^L\right)$
- Vyjdeme z definice a aplikujeme derivaci složené funkce:

$$\delta_j^L \equiv \frac{\partial C}{\partial z_j^L} = \sum_{k=1}^{n_L} \frac{\partial C}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L}$$

přitom aktivace k-tého neuronu poslední vrstvy závisí pouze na $z_k^L \implies \frac{\partial a_k^L}{\partial z_{j^L}} = 0 \text{ pokud } j \neq k$

Backpropagation - 4 rovnice algoritmu: důkaz

- **Důkaz 1. :** $\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'\left(z_j^L\right)$
- Vyjdeme z definice a aplikujeme derivaci složené funkce:

$$\delta_j^L \equiv \frac{\partial C}{\partial z_j^L} = \sum_{k=1}^{n_L} \frac{\partial C}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L}$$

přitom aktivace k-tého neuronu poslední vrstvy závisí pouze na $z_k^L \implies \frac{\partial a_k^L}{\partial z_{j^L}} = 0 \text{ pokud } j \neq k$

Dohromady tedy $\delta_j^L \equiv \frac{\partial C}{\partial z_j^L} = \sum_{k=1}^{n_L} \frac{\partial C}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L}$

Backpropagation - 4 rovnice algoritmu: důkaz

- **Důkaz 1. :** $\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'\left(z_j^L\right)$
- Vyjdeme z definice a aplikujeme derivaci složené funkce:

$$\delta_j^L \equiv \frac{\partial C}{\partial z_j^L} = \sum_{k=1}^{n_L} \frac{\partial C}{\partial a_k^L}\frac{\partial a_k^L}{\partial z_j^L}$$

přitom aktivace k-tého neuronu poslední vrstvy závisí pouze na $z_k^L \implies \frac{\partial a_k^L}{\partial z_{j^L}} = 0 \text{ pokud } j \neq k$

Dohromady tedy $\delta_j^L \equiv \frac{\partial C}{\partial z_j^L} = \sum_{k=1}^{n_L} \frac{\partial C}{\partial a_k^L}\frac{\partial a_k^L}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L}$

Dále víme, že $a_j^L = \sigma\left(z_j^L\right)$, což můžeme dosadit do předchozího vztahu

Backpropagation - 4 rovnice algoritmu: důkaz

- **Důkaz 1. :** $\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'\left(z_j^L\right)$
- Vyjdeme z definice a aplikujeme derivaci složené funkce:

$$\delta_j^L \equiv \frac{\partial C}{\partial z_j^L} = \sum_{k=1}^{n_L} \frac{\partial C}{\partial a_k^L}\frac{\partial a_k^L}{\partial z_j^L}$$

přitom aktivace k-tého neuronu poslední vrstvy závisí pouze na $z_k^L \implies \frac{\partial a_k^L}{\partial z_{j^L}} = 0 \text{ pokud } j \neq k$

Dohromady tedy $\delta_j^L \equiv \frac{\partial C}{\partial z_j^L} = \sum_{k=1}^{n_L} \frac{\partial C}{\partial a_k^L}\frac{\partial a_k^L}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L}$

Dále víme, že $a_j^L = \sigma\left(z_j^L\right)$, což můžeme dosadit do předchozího vztahu

$$\delta_j^L \equiv \frac{\partial C}{\partial z_j^L} = \sum_{k=1}^{n_L} \frac{\partial C}{\partial a_k^L}\frac{\partial a_k^L}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L}\frac{\partial \sigma}{\partial z_j^L}\left(z_j^L\right) = \frac{\partial C}{\partial a_j^L}\sigma'\left(z_j^L\right)$$

Backpropagation - 4 rovnice algoritmu: důkaz

- **Důkaz 2. :** $\delta_j^l = \sum_{k=1}^{n_{l+1}} w_{kj}^{l+1} \delta_k^{l+1} \sigma'\left(z_j^l\right)$

Backpropagation - 4 rovnice algoritmu: důkaz

- **Důkaz 2. :** $\delta_j^l = \sum_{k=1}^{n_{l+1}} w_{kj}^{l+1} \delta_k^{l+1} \sigma'\left(z_j^l\right)$
  - Chceme přepsat chybu v ***l***-té vrstvě pomocí chyby v ***(l+1)*** vrstvě

Backpropagation - 4 rovnice algoritmu: důkaz

- **Důkaz 2. :** $\delta_j^l = \sum_{k=1}^{n_{l+1}} w_{kj}^{l+1} \delta_k^{l+1} \sigma'\left(z_j^l\right)$
  - Chceme přepsat chybu v ***l***-té vrstvě pomocí chyby v ***(l+1)*** vrstvě

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_{k=1}^{n_{l+1}} \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_{k=1}^{n_{l+1}} \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^l}$$

Backpropagation - 4 rovnice algoritmu: důkaz

- **Důkaz 2. :** $\delta_j^l = \sum_{k=1}^{n_{l+1}} w_{kj}^{l+1} \delta_k^{l+1} \sigma'\left(z_j^l\right)$

- Chceme přepsat chybu v ***l***-té vrstvě pomocí chyby v ***(l+1)*** vrstvě

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_{k=1}^{n_{l+1}} \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_{k=1}^{n_{l+1}} \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^l}$$

zároveň z definice ***z***: $z_k^{l+1} = \sum_{j=1}^{n_{l+1}} w_{kj}^{l+1} a_j^l + b_k^{l+1} = \sum_{j=1}^{n_{l+1}} w_{kj}^{l+1} \sigma\left(z_j^l\right) + b_k^{l+1}$

Backpropagation - 4 rovnice algoritmu: důkaz

- **Důkaz 2. :** $\delta_j^l = \sum_{k=1}^{n_{l+1}} w_{kj}^{l+1} \delta_k^{l+1} \sigma'\left(z_j^l\right)$

- Chceme přepsat chybu v ***l***-té vrstvě pomocí chyby v ***(l+1)*** vrstvě

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_{k=1}^{n_{l+1}} \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_{k=1}^{n_{l+1}} \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^l}$$

zároveň z definice ***z***: $z_k^{l+1} = \sum_{j=1}^{n_{l+1}} w_{kj}^{l+1} a_j^l + b_k^{l+1} = \sum_{j=1}^{n_{l+1}} w_{kj}^{l+1} \sigma\left(z_j^l\right) + b_k^{l+1}$

což můžeme zderivovat: $\frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1} \sigma'\left(z_j^l\right)$

Backpropagation - 4 rovnice algoritmu: důkaz

- **Důkaz 2. :** $\delta_j^l = \sum_{k=1}^{n_{l+1}} w_{kj}^{l+1} \delta_k^{l+1} \sigma'\left(z_j^l\right)$

- Chceme přepsat chybu v ***l***-té vrstvě pomocí chyby v ***(l+1)*** vrstvě

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_{k=1}^{n_{l+1}} \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_{k=1}^{n_{l+1}} \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^l}$$

zároveň z definice ***z***: $z_k^{l+1} = \sum_{j=1}^{n_{l+1}} w_{kj}^{l+1} a_j^l + b_k^{l+1} = \sum_{j=1}^{n_{l+1}} w_{kj}^{l+1} \sigma\left(z_j^l\right) + b_k^{l+1}$

což můžeme zderivovat: $\dfrac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1} \sigma'\left(z_j^l\right)$

a dosadit do původní rovnice:

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_{k=1}^{n_{l+1}} \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_{k=1}^{n_{l+1}} \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_{k=1}^{n_{l+1}} \delta_k^{l+1} w_{kj}^{l+1} \sigma'\left(z_j^l\right) = \sum_{k=1}^{n_{l+1}} w_{kj}^{l+1} \delta_k^{l+1} \sigma'\left(z_j^l\right)$$

Backpropagation - 4 rovnice algoritmu: důkaz (cvičení)

- **Důkaz 3. :** $\frac{\partial C}{\partial b_j^l} = \delta_j^l$

Backpropagation - 4 rovnice algoritmu: důkaz (cvičení)

- **Důkaz 3. :** $\frac{\partial C}{\partial b_j^l} = \delta_j^l$
  - Vyjdeme z definice chyby neuronu a použijeme derivaci složené funkce (rozvineme závislost na biasu)

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_{k=1}^{n_{l+1}} \frac{\partial C}{\partial b_k^l} \frac{\partial b_k^l}{\partial z_j^l} = \frac{\partial C}{\partial b_j^l} \frac{\partial b_j^l}{\partial z_j^l}, \text{protože}\, b_k^l \,\text{závisí jen na}\ z_k^l,\ \text{neboli}\ \frac{\partial b_k^l}{\partial z_j^l} = 0\ \text{když j} \neq \text{k}$$

Backpropagation - 4 rovnice algoritmu: důkaz (cvičení)

- **Důkaz 3. :** $\dfrac{\partial C}{\partial b_j^l} = \delta_j^l$
  - Vyjdeme z definice chyby neuronu a použijeme derivaci složené funkce (rozvineme závislost na biasu)

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_{k=1}^{n_{l+1}} \frac{\partial C}{\partial b_k^l} \frac{\partial b_k^l}{\partial z_j^l} = \frac{\partial C}{\partial b_j^l} \frac{\partial b_j^l}{\partial z_j^l}, \text{protože } b_k^l \text{ závisí jen na } z_k^l, \text{ neboli } \frac{\partial b_k^l}{\partial z_j^l} = 0 \text{ když j} \neq \text{k}$$

dále z definice $z_j^l = \sum_{k=1}^{n_l} w_{kj}^l a_k^{l-1} + b_j^l \iff b_j^l = z_j^l - \sum_{k=1}^{n_l} w_{kj}^l a_k^{l-1}, \text{ zderivováním získáme } \dfrac{\partial b_j^l}{z_j^l} = 1$

Backpropagation - 4 rovnice algoritmu: důkaz (cvičení)

- **Důkaz 3. :** $\frac{\partial C}{\partial b_j^l} = \delta_j^l$
  - Vyjdeme z definice chyby neuronu a použijeme derivaci složené funkce (rozvineme závislost na biasu)

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_{k=1}^{n_{l+1}} \frac{\partial C}{\partial b_k^l} \frac{\partial b_k^l}{\partial z_j^l} = \frac{\partial C}{\partial b_j^l} \frac{\partial b_j^l}{\partial z_j^l}, \text{protože } b_k^l \text{ závisí jen na } z_k^l, \text{ neboli } \frac{\partial b_k^l}{\partial z_j^l} = 0 \text{ když j} \neq \text{k}$$

dále z definice $z_j^l = \sum_{k=1}^{n_l} w_{kj}^l a_k^{l-1} + b_j^l \iff b_j^l = z_j^l - \sum_{k=1}^{n_l} w_{kj}^l a_k^{l-1}$, zderivováním získáme $\frac{\partial b_j^l}{z_j^l} = 1$

což dosadíme do předchozí rovnice: $\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_{k=1}^{n_{l+1}} \frac{\partial C}{\partial b_k^l} \frac{\partial b_k^l}{\partial z_j^l} = \frac{\partial C}{\partial b_j^l} \frac{\partial b_j^l}{\partial z_j^l} = \frac{\partial C}{\partial b_j^l}$

Backpropagation - 4 rovnice algoritmu: důkaz (cvičení)

- **Důkaz 4. :** $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$

Backpropagation - 4 rovnice algoritmu: důkaz (cvičení)

- **Důkaz 4. :** $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$
- Podobné úvahy jako předtím, použijeme derivaci složené funkce (rozvineme **z**) a uvědomíme si, že z^l_j závisí pouze na w^l_{jk}, takže se všechny členy sumy kromě i=j vynulují

Backpropagation - 4 rovnice algoritmu: důkaz (cvičení)

- **Důkaz 4. :** $\dfrac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$

- Podobné úvahy jako předtím, použijeme derivaci složené funkce (rozvineme **z**) a uvědomíme si, že z^l_j závisí pouze na w^l_{jk}, takže se všechny členy sumy kromě i=j vynulují

$$\frac{\partial C}{\partial w^l_{jk}} = \sum_{i=1}^{n_l} \frac{\partial C}{\partial z^l_i} \frac{\partial z^l_i}{\partial w^l_{jk}} = \frac{\partial C}{\partial z^l_j} \frac{\partial z^l_j}{\partial w^l_{jk}} = \delta^l_j \frac{\partial z^l_j}{\partial w^l_{jk}} = \delta^l_j a^{l-1}_k, \text{ protože z definice } z^l_j = \sum_{k=1}^{n_l} w^l_{jk} a^{l-1}_k \odot + b^l_j,$$

$$\text{a tedy } \frac{\partial z^l_j}{\partial w^l_{jk}} = a^{l-1}_k$$

Backpropagation - cvičení

- Cvičení: Odvoďte maticovou formu první a druhé rovnice

Backpropagation - cvičení

- Cvičení: Odvoďte maticovou formu první a druhé rovnice
- První rovnici můžeme přepsat pomocí standardního maticového počtu:

$$\delta^L = \nabla_a C \odot \sigma'\left(z^L\right) = \begin{pmatrix} \frac{\partial C}{\partial a_1^L} \\ \vdots \\ \frac{\partial C}{\partial a_{n_L}^L} \end{pmatrix} \odot \begin{pmatrix} \sigma'\left(z_1^L\right) \\ \vdots \\ \sigma'\left(z_{n_L}^L\right) \end{pmatrix} = \begin{pmatrix} \sigma'\left(z_1^L\right) \\ \vdots \\ \sigma'\left(z_{n_L}^L\right) \end{pmatrix} \odot \begin{pmatrix} \frac{\partial C}{\partial a_1^L} \\ \vdots \\ \frac{\partial C}{\partial a_{n_L}^L} \end{pmatrix} =$$

$$= \begin{pmatrix} \sigma'\left(z_1^L\right) & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & \sigma'\left(z_{n_L}^L\right) \end{pmatrix} \begin{pmatrix} \frac{\partial C}{\partial a_1^L} \\ \vdots \\ \frac{\partial C}{\partial a_{n_L}^L} \end{pmatrix} = \Sigma'\left(z^L\right) \nabla_a C$$

Backpropagation - cvičení

- Cvičení: Odvoďte maticovou formu první a druhé rovnice
- První rovnici můžeme přepsat pomocí standardního maticového počtu:

$$\delta^L = \nabla_a C \odot \sigma'(z^L) = \begin{pmatrix} \frac{\partial C}{\partial a_1^L} \\ \vdots \\ \frac{\partial C}{\partial a_{n_L}^L} \end{pmatrix} \odot \begin{pmatrix} \sigma'(z_1^L) \\ \vdots \\ \sigma'(z_{n_L}^L) \end{pmatrix} = \begin{pmatrix} \sigma'(z_1^L) \\ \vdots \\ \sigma'(z_{n_L}^L) \end{pmatrix} \odot \begin{pmatrix} \frac{\partial C}{\partial a_1^L} \\ \vdots \\ \frac{\partial C}{\partial a_{n_L}^L} \end{pmatrix} =$$

$$= \begin{pmatrix} \sigma'(z_1^L) & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & \sigma'(z_{n_L}^L) \end{pmatrix} \begin{pmatrix} \frac{\partial C}{\partial a_1^L} \\ \vdots \\ \frac{\partial C}{\partial a_{n_L}^L} \end{pmatrix} = \Sigma'(z^L) \nabla_a C$$

- Stejně můžeme upravit i druhou rovnici:

$$\delta^l = \left(\left(w^{l+1}\right)^T \delta^{l+1} \right) \odot \sigma'(z^l) = \Sigma'(z^l) \left(w^{l+1}\right)^T \delta^{l+1}$$

Backpropagation - cvičení

- Cvičení: Dokažte vztah $\delta^l = \Sigma'(z^l)\left(w^{l+1}\right)^T \cdots \Sigma'(z^{L-1})\left(w^L\right)^T \Sigma'(z^L)\nabla_a C$
- Budeme-li v druhé rovnici rekurentně dosazovat za deltu až po ***l+1=L***, získáme

$$
\begin{aligned}
\delta^l &= \Sigma'(z^l)\left(w^{l+1}\right)^T \delta^{l+1} = \Sigma'(z^l)\left(w^{l+1}\right)^T \Sigma'(z^{l+1})\left(w^{l+2}\right)^T = \cdots = \\
&= \Sigma'(z^l)\left(w^{l+1}\right)^T \cdots \Sigma'(z^{L-1})\left(w^{L-1}\right)^T \Sigma'(z^L)\left(w^L\right)^T = \\
&= \Sigma'(z^l)\left(w^{l+1}\right)^T \cdots \Sigma'(z^{L-1})\left(w^L\right)^T \Sigma'(z^L)\nabla_a C
\end{aligned}
$$

Backpropagation - popis algoritmu

1. Input x: nastavíme aktivaci input vrstvy a^1
2. Feedforward: pro $l = 2, 3, \ldots, L$ spočítáme $z^l = w^l a^{l-1} + b^l$ a $a^l = \sigma(z^l)$
3. Output error: spočítáme vektor $\delta^L = \nabla_a C \odot \sigma'(z^L)$
4. Backpropagation: pro $l = L-1, L-2, \ldots, 2$ spočítáme $\delta^l = \left(\left(w^{l+1}\right)^T \delta^{l+1}\right) \odot \sigma'(z^l)$
5. Output: gradient účelové funkce je dán vztahy $\frac{\partial C}{\partial w^l_{jk}} = {a_k}^{l-1} \delta^l$ a $\frac{\partial C}{\partial b^l_j} = \delta^l_j$

Backpropagation - popis algoritmu

1. Input x: nastavíme aktivaci input vrstvy a^1
2. Feedforward: pro $l = 2, 3, \ldots, L$ spočítáme $z^l = w^l a^{l-1} + b^l$ a $a^l = \sigma(z^l)$
3. Output error: spočítáme vektor $\delta^L = \nabla_a C \odot \sigma'(z^L)$
4. Backpropagation: pro $l = L-1, L-2, \ldots, 2$ spočítáme $\delta^l = \left(\left(w^{l+1}\right)^T \delta^{l+1}\right) \odot \sigma'(z^l)$
5. Output: gradient účelové funkce je dán vztahy $\frac{\partial C}{\partial w^l_{jk}} = a_k{}^{l-1} \delta^l$ a $\frac{\partial C}{\partial b^l_j} = \delta^l_j$

- Chybu počítáme od poslední vrstvy dopředu, proto **backpropagation**

Backpropagation - popis algoritmu

1. Input x: nastavíme aktivaci input vrstvy a^1
2. Feedforward: pro $l = 2, 3, \ldots, L$ spočítáme $z^l = w^l a^{l-1} + b^l$ a $a^l = \sigma(z^l)$
3. Output error: spočítáme vektor $\delta^L = \nabla_a C \odot \sigma'(z^L)$
4. Backpropagation: pro $l = L-1, L-2, \ldots, 2$ spočítáme $\delta^l = \left(\left(w^{l+1}\right)^T \delta^{l+1}\right) \odot \sigma'(z^l)$
5. Output: gradient účelové funkce je dán vztahy $\frac{\partial C}{\partial w^l_{jk}} = a_k{}^{l-1} \delta^l$ a $\frac{\partial C}{\partial b^l_j} = \delta^l_j$

- Chybu počítáme od poslední vrstvy dopředu, proto **backpropagation**
- to vychází z toho, jak je účelová funkce závislá na parametrech sítě (aktivace vrstvy je závislá na aktivacích předchozí vrstvy)

Backpropagation - popis algoritmu

1. Input x: nastavíme aktivaci input vrstvy a^1
2. Feedforward: pro $l = 2, 3, \ldots, L$ spočítáme $z^l = w^l a^{l-1} + b^l$ a $a^l = \sigma(z^l)$
3. Output error: spočítáme vektor $\delta^L = \nabla_a C \odot \sigma'(z^L)$
4. Backpropagation: pro $l = L-1, L-2, \ldots, 2$ spočítáme $\delta^l = \left(\left(w^{l+1}\right)^T \delta^{l+1}\right) \odot \sigma'(z^l)$
5. Output: gradient účelové funkce je dán vztahy $\frac{\partial C}{\partial w^l_{jk}} = a_k{}^{l-1}\, \delta^l$ a $\frac{\partial C}{\partial b^l_j} = \delta^l_j$

- Chybu počítáme od poslední vrstvy dopředu, proto **backpropagation**
- to vychází z toho, jak je účelová funkce závislá na parametrech sítě (aktivace vrstvy je závislá na aktivacích předchozí vrstvy)
- teď už máme vše potřebné k naprogramování neuronky

Backpropagation - popis algoritmu

1. Input x: nastavíme aktivaci input vrstvy a^1
2. Feedforward: pro $l = 2, 3, \ldots, L$ spočítáme $z^l = w^l a^{l-1} + b^l$ a $a^l = \sigma(z^l)$
3. Output error: spočítáme vektor $\delta^L = \nabla_a C \odot \sigma'(z^L)$
4. Backpropagation: pro $l = L-1, L-2, \ldots, 2$ spočítáme $\delta^l = \left(\left(w^{l+1}\right)^T \delta^{l+1}\right) \odot \sigma'(z^l)$
5. Output: gradient účelové funkce je dán vztahy $\frac{\partial C}{\partial w^l_{jk}} = a_k{}^{l-1} \delta^l$ a $\frac{\partial C}{\partial b^l_j} = \delta^l_j$

- Chybu počítáme od poslední vrstvy dopředu, proto **backpropagation**
- to vychází z toho, jak je účelová funkce závislá na parametrech sítě (aktivace vrstvy je závislá na aktivacích předchozí vrstvy)
- teď už máme vše potřebné k naprogramování neuronky
- algoritmus se dá ještě vylepšit pro mini batch, aby místo cyklu použil maticové násobení (ukážeme si v kódu)

Backpropagation - cvičení

- Představte si, že změníte jeden neuron v síti, konkrétně mu přiřadíte jinou aktivační funkci ***f***. Jak se změní backpropagation

Backpropagation - cvičení

- Představte si, že změníte jeden neuron v síti, konkrétně mu přiřadíte jinou aktivační funkci ***f***. Jak se změní backpropagation
  - v podstatě se nezmění nic, jen kromě derivace sigmoid funkce bude u příslušného neuronu v rovnicích derivace ***f***

Backpropagation - cvičení

- Představte si, že změníte jeden neuron v síti, konkrétně mu přiřadíte jinou aktivační funkci ***f***. Jak se změní backpropagation
  - v podstatě se nezmění nic, jen kromě derivace sigmoid funkce bude u příslušného neuronu v rovnicích derivace ***f***
  - z toho vidíme, že náš algoritmus nezávisí na volbě aktivační funkce (za splnění jistých předpokladů)

Backpropagation - cvičení

- Představte si, že nahradíte sigmoid identitou. Přepište rovnice backpropagation algoritmu

Backpropagation - závěr

- Backpropagation je ukrutně rychlý: představte si alternativní přístup, kdy chcete derivaci účelové funkce aproximovat $\frac{\partial C}{\partial w_j} \approx \frac{C(w + \epsilon e_j) - C(w)}{\epsilon}$

Backpropagation - závěr

- Backpropagation je ukrutně rychlý: představte si alternativní přístup, kdy chcete derivaci účelové funkce aproximovat $\frac{\partial C}{\partial w_j} \approx \frac{C(w + \epsilon e_j) - C(w)}{\epsilon}$
- to vyžaduje provést forward pass pro každou modifikovanou váhu a originální váhy v síti. Oproti tomu backpropagation potřebuje 1 forward pass a 1 backward pass, který je stejně výpočetně náročný.

Backpropagation - závěr

- Backpropagation je ukrutně rychlý: představte si alternativní přístup, kdy chcete derivaci účelové funkce aproximovat $\frac{\partial C}{\partial w_j} \approx \frac{C(w+\epsilon e_j) - C(w)}{\epsilon}$
- to vyžaduje provést forward pass pro každou modifikovanou váhu a originální váhy v síti. Oproti tomu backpropagation potřebuje 1 forward pass a 1 backward pass, který je stejně výpočetně náročný.
- Pro síť s milionem vah je to rozdíl 1 000 001 ku 2.