Neurovõrgud. Praktikum 3.

Ülesannete lahendused

Ülesanne 1: Näita, et aktivatsioonifunktsiooniga ϕ neuroni online-treenimise samm on

$$\Delta \mathbf{w}_k = \eta\phi'(\mathbf{w}^T\mathbf{x} + b)e(i)\mathbf{x}(i) \qquad \Delta b = \eta\phi'(\mathbf{w}^T\mathbf{x} + b)e(i)$$

ning juhul kui $\phi(x) = \tanh(x)$

$$\Delta \mathbf{w}_k = \eta(1 - y^2(i))e(i)\mathbf{x}(i) \qquad \Delta b = \eta(1 - y^2(i)e(i)$$

Lahendus: Minimiseeritav veafunktsioon on $\mathcal{E} = \sum_i e^2(i)$. Kuna me kasutame online-algoritmi, valime igal sammul juhuslikult treeningnäite i ning minimiseerime võrgu vea ainult sellel näitel:

$$\mathcal{E}(\mathbf{w}, b) = e^2(i) = (d(i) - \phi(\mathbf{w}^T\mathbf{x}(i) + b))^2$$

Algoritmi samm on siis funktsiooni $\mathcal{E}$ gradiendile vastu:

$$\begin{aligned}
\Delta\mathbf{w} &= -\eta\frac{\partial\mathcal{E}}{\partial\mathbf{w}} = -\eta\frac{\partial e^2(i)}{\partial\mathbf{w}} = -2\eta e(i)\frac{\partial(d(i) - \phi(\mathbf{w}^T\mathbf{x}(i) + b))}{\partial\mathbf{w}} = \\
&= 2\eta e(i)\phi'(\mathbf{w}^T\mathbf{x}(i) + b)\mathbf{x}(i) \\
\Delta b &= -\eta\frac{\partial\mathcal{E}}{\partial b} = -\eta\frac{\partial e^2(i)}{\partial b} = -2\eta e(i)\frac{\partial(d(i) - \phi(\mathbf{w}^T\mathbf{x}(i) + b))}{\partial b} = \\
&= 2\eta e(i)\phi'(\mathbf{w}^T\mathbf{x}(i) + b)
\end{aligned}$$

Konstandi 2 saab välja visata ning esimene väide on tõestatud. Nüüd kui $\phi(x) = \tanh(x)$, siis

$$\begin{aligned}
\phi'(x) = \tanh'(x) &= \frac{\partial}{\partial x}\frac{e^x - e^{-x}}{e^x + e^{-x}} = \\
&= \frac{(e^x + e^{-x})^2 - (e^x - e^{-x})^2}{(e^x + e^{-x})^2} = 1 - \tanh^2(x)
\end{aligned}$$

millest järeldub teise väide tõesus.

Ülesanne 3: Miks nulli eristamiseks treenitud neuroni kaalude pilt sarnaneb nulliga?

Lahendus: Siin on võimalikud erinevad seletused, näiteks:

- Neuroni väljund peab olema võimalikult positiivne nulli puhul, ning võimalikult negatiivne teiste piltide puhul. Seega kaalud, mis vastavad nulli pikslitele on positiivsed, ning kaalud, mis vastavad teiste piltide pikslitele on negatiivsed. Muud kaalud on nullilähedased.
- Neuroni kaalude vektor peab olema positiivselt korreleeritud nulli piltidega. Selline vektor on näiteks nende piltide keskväärtus.
- Kujutame ette eraldava hüpertasandi. Selle tasandi normaal on suunatud nullide poole. Neuroni kaalude vektor ongi see normaal.

Ülesanne 4: Pane kirja vea tagastamise algoritmi sammud maatrikskujul.

Lahendus: Tähistagu $v^{(s)}$, $y^{(s)}$, $\delta^{(s)}$ s-nda kihi neuronite lokaalse välja vektorit, väljundite vektorit ning lokaalsete gradientide vektorit vastavalt. Olgu $\mathbf{W}^{(s)}$ — kihi s kaalude maatriks, kus $w_{ji}^{(s)}$ on kaal kihi s neuroni j ja kihi $s-1$ neuroni i vahel. Siis edaspidises läbimises:

$$v^{(s+1)} = \hat{\mathbf{W}}^{(s+1)}\hat{y}^{(s)}$$

kus $\hat{\mathbf{W}}^{(s+1)}$ on maatriks $\mathbf{W}^{(s+1)}$ kuhu ette oli lisatud kihi $s+1$ neuronite bias-parameetrite veerg, ning $\hat{y}^{(s)}$ on vektor $y^{(s)}$ kuhu ette oli lisatud element 1.
Äraspidises läbimises:

$$\hat{\delta}^{(s)} = \hat{\mathbf{W}}^{(s+1)T}\delta^{(s+1)}$$

kus vektori $\hat{\delta}^{(s)}$ esimene komponent on kihi s bias-neuroni lokaalne gradient.

Ülesanne 6: Kui palju elemente on 1000-100-1 võrgu veafunktsiooni hessiaanis?

Lahendus: n-parameetriga funktsiooni f hessiaan on funktsiooni f kõikvõimalike teiste tuletiste maatriks suurusega $n \times n$. Seega ülesande lahendamiseks peame loendama, kui palju parameetreid on antud võrgu veafunktsioonil, ehk kui palju kaalu on selles võrgus.
Sisendkihi ja peidetud kihi vahel on 1000×100 kaalu, peidetud ja väljundkihi vahel on 100 kaalu. Lisaks on võrgu igal mitte-sisend-neuronil olemas bias, mis on sisuliselt ka kaal. Neid on $100 + 1$. Niiet kokku on võrgus

$$1000 \times 100 + 100 + 1 = 100101$$

kaalu.
Hessiaani elemente on seega $100101^2 = 10\ 020\ 210\ 201$. Kui iga arvu salvestamiseks on vaja 4 baiti, ning on arvestatatud hessiaani sümmeetrilisus, nõuab sellise maatriksi hoidmine üle 18 gigabaiti.

Ülesanne: Kuidas sõltub õppimiskõver võrgu neuronite arvust?

Lahendus: Selles ülesandes oli vaja panna tähele asjaolu, et mida rohkem on võrgus neuroneid, seda suuremad on õppimiskõvera kõikumised. See on loomulik, kui mõelda sellele kuidas vea tagastamise algoritm töötab: iga neuroni kaalude muut on proportsionaalne selle neuroni lokaalsele väljale, mis on järgmise kihi lokaalsete väljade kaalutud summa. Seega kui järgmises kihis on palju neuroneid, on tõenäoline et summerides nende lokaalseid gradiente saame suure arvu. Sellest tulenevad suured muutused kaaludele, ning veafunktsiooni suured kõikumised.
Suurte kõikumistega õppimiskõver on halb ning ei anna head hinnangut sellele, kui hästi veafunktsioon minimiseeritud sai. Tõepoolest, kui kõver kõikub igal sammul näiteks 10 ja 1000 vahel, siis see fakt, et mingil sammul on tal väärtus 10 ei tähenda seda et võrk on nüüd hästi treenitud. See arv 10 oli mingis mõttes juhuslik ning kohe järgmisel sammul võib see jälle muutuda 1000-ks.
Selleks et õppimiskõver oleks sile tuleb vähendada õppimisteguri η. Väga väikse η puhul on aga treenimine aeglane. Saab panna tähele, et väiksem η peab olema ainult nendel kaaludel, mis on seotud suure lokaalse gradiendiga neuroniga (sest ainult nendel kaaludel võib muutus olla liiga suur). Seega õppimist saab optimeerida valides igale kaalule temale spetsiifilist kaalu. See on nn. *quasi-newton*-treenimismeetodite mõte.
Veel üks võimalik lahendus on teha alati fikseeritud pikkusega samme gradiendi vastassuunas. S.t. niimoodi et vektori $\Delta\mathbf{w}$ norm oleks alati η. Sellist algoritmi nimetatakse *R-propagation*.