ML HW 8 - Deep Learning 2

Monday, January 18, 2021 10:56 AM

Homework

Problem 2: You are trying to solve a regression task and you want to choose between two approaches:

1. A simple linear regression model.
2. A feed forward neural network $f_{\boldsymbol{W}}(\boldsymbol{x})$ with L hidden layers, where each hidden layer $l \in \{1, ..., L\}$ has a weight matrix $\boldsymbol{W}_l \in \mathbb{R}^{D\times D}$ and a ReLU activation function. The output layer has a weight matrix $\boldsymbol{W}_{L+1} \in \mathbb{R}^{D\times 1}$ and no activation function.

In both models, there are no bias terms.

Your dataset $\mathcal{D}$ contains data points with nonnegative features $\boldsymbol{x}_n$ and the target y_n is continuous:

$$\mathcal{D} = \{\boldsymbol{x}_n, y_n\}_{n=1}^N, \qquad \boldsymbol{x}_n \in \mathbb{R}^D_{\geq 0}, \qquad y_n \in \mathbb{R}$$

Let $\boldsymbol{w}^*_{LS} \in \mathbb{R}^D$ be the optimal weights for the linear regression model corresponding to a *global* minimum of the following least squares optimization problem:

$$\boldsymbol{w}^*_{LS} = \arg\min_{\boldsymbol{w}\in\mathbb{R}^D} \mathcal{L}_{LS}(\boldsymbol{w}) = \arg\min_{\boldsymbol{w}\in\mathbb{R}^D} \frac{1}{2}\sum_{n=1}^N (\boldsymbol{w}^T\boldsymbol{x}_n - y_n)^2$$

Let $\boldsymbol{W}^*_{NN} = \{\boldsymbol{W}^*_1, \ldots, \boldsymbol{W}^*_{L+1}\}$ be the optimal weights for the neural network corresponding to a *global* minimum of the following optimization problem:

$$\boldsymbol{W}^*_{NN} = \arg\min_{\boldsymbol{W}} \mathcal{L}_{NN}(\boldsymbol{W}) = \arg\min_{\boldsymbol{W}} \frac{1}{2}\sum_{n=1}^N (f_{\boldsymbol{W}}(\boldsymbol{x}_n) - y_n)^2$$

a) Assume that the optimal $\boldsymbol{W}^*_{NN}$ you obtain are non-negative.
What will the relation ($<, \leq, =, \geq, >$) between the neural network loss $\mathcal{L}_{NN}(\boldsymbol{W}^*_{NN})$ and the linear regression loss $\mathcal{L}_{LS}(\boldsymbol{w}^*_{LS})$ be? Provide a mathematical argument to justify your answer.

> Note that for any non-negative $\boldsymbol{x}$ and any non-negative $\boldsymbol{W}$ it holds $\mathrm{ReLU}(\boldsymbol{x}\boldsymbol{W}) = \boldsymbol{x}\boldsymbol{W}$.
>
> Therefore, since our data points have non-negative features $\boldsymbol{x}_i$ and the optimal weights $\boldsymbol{W}^*_{NN}$ are non-negative, every ReLU layer is equivalent to a linear layer when plugging in the optimal weights. This means we can write
>
> $$\begin{aligned} f_{\boldsymbol{W}^*_{NN}}(\boldsymbol{x}_i) &= ReLU(ReLU(ReLU(\boldsymbol{x}_i^T\boldsymbol{W}^*_1)\boldsymbol{W}^*_2)\cdots\boldsymbol{W}^*_L)\boldsymbol{W}^*_{L+1} \\ &= \boldsymbol{x}_i^T\boldsymbol{W}^*_1\boldsymbol{W}^*_2\cdots\boldsymbol{W}^*_{L+1} \\ &= \boldsymbol{x}_i^T\boldsymbol{w}^*_{NN} \end{aligned}$$
>
> where we defined $\boldsymbol{w}^*_{NN} = \boldsymbol{W}^*_1\boldsymbol{W}^*_2\cdots\boldsymbol{W}^*_{L+1}$. From this we can see that the neural network with optimal weights behaves like a linear regression with a different set of weights $\boldsymbol{w}^*_{NN}$.
>
> Note also that linear regression is a special case of the above neural network, i.e. for any weights $\boldsymbol{w}_{LS}$ you can find weights $\boldsymbol{W}_{NN}$ that produce the same output.
>
> Given the above facts and since we the optimal weights correspond to a global minima we can conclude that $\mathcal{L}_{NN}(\boldsymbol{W}^*_{NN}) = \mathcal{L}_{LS}(\boldsymbol{w}^*_{LS})$ and the optimal weights found by solving the least squares optimization problem will be $\boldsymbol{w}^*_{LS} = \boldsymbol{w}^*_{NN}$.

b) In contrast to (a), now assume that the optimal weights $\boldsymbol{w}^*_{LS}$ you obtain are non-negative.
What will the relation ($<, \leq, =, \geq, >$) between the linear regression loss $\mathcal{L}_{LS}(\boldsymbol{w}^*_{LS})$ and the neural network loss $\mathcal{L}_{NN}(\boldsymbol{W}^*_{NN})$ be? Provide a mathematical argument to justify your answer.

> As stated in (a) linear regression is a special case of the above neural network, i.e. for any weights $\boldsymbol{w}_{LS}$ you can find weights $\boldsymbol{W}_{NN}$ that produce the same output. That is, everything that can be learned with a linear regression can be learned equally well with a neural network.
>
> However, the reverse direction doesn't hold, since in principle neural networks can learn more complicated functions compared to linear regression. Moreover, the given fact that $\boldsymbol{w}^*_{LS}$ are non-negative does not tell us anything about the optimal weights of the neural network $\boldsymbol{W}^*_{NN}$.
>
> Therefore it holds $\mathcal{L}_{NN}(\boldsymbol{W}^*_{NN}) \leq \mathcal{L}_{LS}(\boldsymbol{w}^*_{LS})$ since the neural network can potentially find a better fit for the data (e.g. by taking advantage of non-linearity).

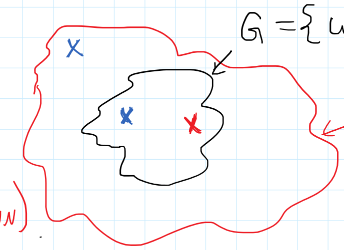
Any LR model $g(x) = w^Tx$ I can represent with a NN model $f_W(x)$.

$f_W(x) = \mathrm{ReLU}(\mathrm{ReLU}(x^T I) I) \cdot w = x^Tw = w^Tx$

$\mathrm{ReLU}(Wx) = Wx$ for $W \geq 0,\ x \geq 0$

Not every NN I can represent with a LR model

$G = \{w^Tx : w \in \mathbb{R}^D\}$



$\mathcal{F} = \{f_W(x) : W \in ..\}$

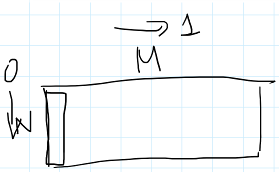
$\mathcal{L}_{LS}(w^*_{LS}) = \mathcal{L}_{NN}(W^*_{NN})$

$G \subset \mathcal{F}$

$$\min_{f\in G} \frac{1}{2}\sum_{n=1}^N (f(x_n) - y_n)^2 \geq \min_{f\in\mathcal{F}} \frac{1}{2}\sum_{n=1}^N (f(x_n) - y_n)^2$$

If $W^*_{NN} \geq 0$, then $f_W(x) = w^Tx$

$\mathcal{L}_{LS}(w^*_{LS}) \geq \mathcal{L}_{NN}(W^*_{NN})$



$= X$

$X.\mathrm{mean}(dim=1)$ N

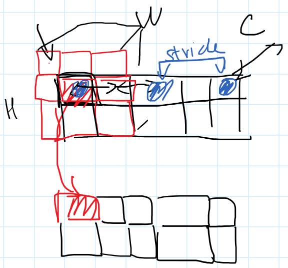
$X.\mathrm{mean}(dim=1,\ keepdim=True)$ N 1

$$\frac{d}{dx}\frac{1}{\sqrt{x+\varepsilon}}$$

X $[N, C, H, W]$

$R(x) = \mathcal{F}(x) + x$

in_channels
out_channels
kernel_size = 3
stride = 1



$X[0, 1, :, :]$ → $[H, W]$

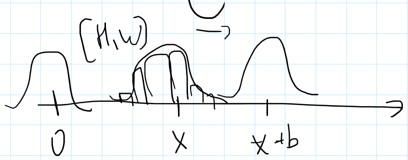
output $[H, W]$

$[N, C, H, W]$

C_{in}, } provided by user
C_{out}

$K = 3$
padding = 1

$$\text{stride} = \begin{cases} 1 \Rightarrow \text{out } [N, C, H, W] \\ 2 \Rightarrow \text{out } [N, C, \frac{H}{2}, \frac{W}{2}] \end{cases}$$



$[N, C, H, W] \leftarrow X$

$[N, C, \frac{H}{2}, \frac{W}{2}] \leftarrow \mathcal{F}(x)$

output shape

$$\begin{array}{cccc} & C & H & W \\ [N, & 3, & 32, & 32] \end{array}$$

conv (k=3, s=1) [N, 16, 32, 32]

Stack $\begin{pmatrix} C_{in}=16, & k=3, \\ C_{out}=16, & S=1 \end{pmatrix}$ [N, 16, 32, 32]

Stack $\begin{pmatrix} C_{in}=16, & k=3, \\ C_{out}=32, & S=2 \end{pmatrix}$ [N, 32, 16, 16]

Stack $\begin{pmatrix} C_{in}=32, & k=3, \\ C_{out}=64, & S=2 \end{pmatrix}$ [N, 64, 8, 8]

def stack (n):

n { ResidualBlock(..)
⋮
ResidualBlock(..)

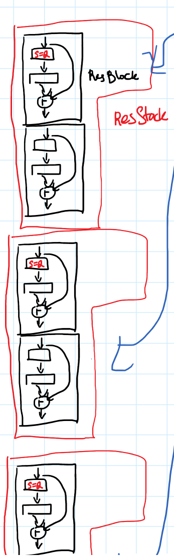
ResBlock:
(S=2) Conv1
conv2
} 2 layers

ResStack (S=2)
ResBlock1
⋮
ResBlock n
} 2n conv. layers = n · ResBlocks

C H W

X [N, 3, 32, 32]

conv (k=3, s=1) [N, 16, 32, 32]



Stack $\begin{pmatrix} C_{in}=16, & k=3, \\ C_{out}=16, & S=1 \end{pmatrix}$ [N, 16, 32, 32]

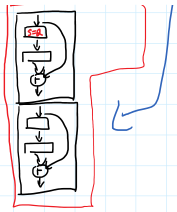
Stack $\begin{pmatrix} C_{in}=16, & k=3, \\ C_{out}=32, & S=2 \end{pmatrix}$ [N, 32, 16, 16]

Stack $\begin{pmatrix} C_{in}=32, & k=3, \\ C_{out}=64, & S=2 \end{pmatrix}$ [N, 64, 8, 8]

AvgPool [N, 64, 1]

squeeze [N, 64]

Linear [N, 10]



Linear [N, 10]

1 conv layer

6n conv. Layers

1 fully connected