Assignment 3

FMAN45 - Machine Learning

Author:
Dahlberg, Vilmer
vi8808da-s@student.lu.se

May 11, 2021

Exercise 1

Derive expressions for $\frac{\partial L}{\partial \boldsymbol{x}}$, $\frac{\partial L}{\partial \boldsymbol{W}}$ and $\frac{\partial L}{\partial \boldsymbol{b}}$ in terms of $\frac{\partial L}{\partial \boldsymbol{z}}$, $\boldsymbol{W}$ and $\boldsymbol{x}$. Include a full derivation of your results. The answers should all be given as matrix expressions without any explicit sums.

Solution

The relationship between $\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{x}$ and $\boldsymbol{z}$ is given by

$$z_i = \sum_{j=1}^{m} W_{ij} x_j + b_i.$$

Begin by computing some derivatives of z

$$\frac{\partial z_l}{\partial x_i} = W_{li} \qquad \frac{\partial z_l}{\partial W_{ij}} = \delta_{il} x_j \qquad \frac{\partial z_l}{\partial b_i} = \delta_{li}. \tag{1}$$

Using the chain rule we have

$$\frac{\partial L}{\partial x_i} = \sum_{l=1}^{n} \frac{\partial L}{\partial z_l} \frac{\partial z_l}{\partial x_i}, \qquad \frac{\partial L}{\partial W_{ij}} = \sum_{l=1}^{n} \frac{\partial L}{\partial z_l} \frac{\partial z_l}{\partial W_{ij}}, \qquad \frac{\partial L}{\partial b_i} = \sum_{l=1}^{n} \frac{\partial L}{\partial z_l} \frac{\partial z_l}{\partial b_i} \tag{2}$$

Insertion of equations 1 into equations 2 gives

$$\frac{\partial L}{\partial x_i} = \sum_{l=1}^{n} \frac{\partial L}{\partial z_l} W_{li} \qquad \frac{\partial L}{\partial W_{ij}} = \sum_{l=1}^{n} \frac{\partial L}{\partial z_l} \delta_{il} x_j = \frac{\partial L}{\partial z_i} x_j \qquad \frac{\partial L}{\partial b_i} = \frac{\partial L}{\partial z_i}$$

The expressions can be rewritten in vectorized form as

$$\frac{\partial L}{\partial \boldsymbol{x}} = \boldsymbol{W}^T \frac{\partial L}{\partial \boldsymbol{z}} \qquad \frac{\partial L}{\partial \boldsymbol{W}} = \frac{\partial L}{\partial \boldsymbol{z}} \boldsymbol{x}^T \qquad \frac{\partial L}{\partial \boldsymbol{b}} = \frac{\partial L}{\partial \boldsymbol{z}}. \tag{3}$$

Exercise 2

Derive expressions for $\boldsymbol{Z}$, $\frac{\partial L}{\partial \boldsymbol{X}}$, $\frac{\partial L}{\partial \boldsymbol{W}}$, and $\frac{\partial L}{\partial \boldsymbol{b}}$ in terms of $\frac{\partial L}{\partial \boldsymbol{Z}}$, $\boldsymbol{W}$, $\boldsymbol{b}$ and $\boldsymbol{X}$. Include a full derivation of your results. Also add MATLAB code that implements these vectorised expressions that you have just derived.

Solution

Begin by deriving an expression for $\boldsymbol{Z}$, which can be written as the below if we let $\boldsymbol{B} = \big(\boldsymbol{b}, \quad \boldsymbol{b}, \quad \boldsymbol{b}...\big)$.

$$\boldsymbol{Z} = \boldsymbol{W}\boldsymbol{X} + \boldsymbol{B}$$

Next, consider the derivatives $\frac{\partial L}{\partial \boldsymbol{X}}$, $\frac{\partial L}{\partial \boldsymbol{W}}$, and $\frac{\partial L}{\partial \boldsymbol{b}}$ using the chain rule and the expressions from the previous exercise,

$$\frac{\partial L}{\partial \boldsymbol{X}} = \left(\frac{\partial L}{\partial \boldsymbol{x}^1}, \ \frac{\partial L}{\partial \boldsymbol{x}^2}, \ ..., \ \frac{\partial L}{\partial \boldsymbol{x}^n}\right) = \left(\boldsymbol{W}^T\frac{\partial L}{\partial \boldsymbol{z}^1}, \ \boldsymbol{W}^T\frac{\partial L}{\partial \boldsymbol{z}^2}, \ ..., \ \boldsymbol{W}^T\frac{\partial L}{\partial \boldsymbol{z}^n}\right) = \boldsymbol{W}^T\frac{\partial L}{\partial \boldsymbol{Z}}$$

$$\frac{\partial L}{\partial W_{ij}} = \sum_{l=1}^{N}\sum_{k=1}^{n}\frac{\partial L}{\partial z_k^l}\frac{\partial z_k^l}{\partial W_{ij}} = \sum_{l=1}^{N}\sum_{k=1}^{n}\frac{\partial L}{\partial z_k^l}\delta_{ki}x_j^l = \sum_{l=1}^{N}\frac{\partial L}{\partial z_i^l}x_j^l \iff \frac{\partial L}{\partial \boldsymbol{W}} = \sum_{l=1}^{N}\frac{\partial L}{\partial \boldsymbol{z}^l}(\boldsymbol{x}^l)^T = \frac{\partial L}{\partial \boldsymbol{Z}}\boldsymbol{X}^T$$

$$\frac{\partial L}{\partial b_i} = \sum_{l=1}^{N}\sum_{k=1}^{n}\frac{\partial L}{\partial z_k^l}\frac{\partial z_k^l}{\partial b_i} = \sum_{l=1}^{N}\sum_{k=1}^{n}\frac{\partial L}{\partial z_k^l}\delta_{ki} = \sum_{l=1}^{N}\frac{\partial L}{\partial z_i^l} \iff \frac{\partial L}{\partial \boldsymbol{b}} = \sum_{l=1}^{N}\frac{\partial L}{\partial \boldsymbol{z}^l} = \frac{\partial L}{\partial \boldsymbol{Z}}\mathbb{1}^{(N\times 1)},$$

where $\mathbb{1}^{(N\times 1)}$ is a $N \times 1$ vector with ones.

Listing 1: `fully_connected_forward`

```
...
    Y = A*X + b;
end
```

Listing 2: `fully_connected_backward`

```
...
    dldX = A'*dldY;
    dldX = reshape(dldX, sz);

    dldA = dldY*X';
    dldb = sum(dldY, 2);

end
```

Exercise 3

Derive the backpropagation expression for $\frac{\partial L}{\partial x_i}$, where $z_i = \max(x_i, 0)$.

Solution

The backpropagation expression can be computed using the chain rule, or

$$\frac{\partial L}{\partial x_i} = \frac{\partial L}{\partial z_i}\frac{\partial z_i}{\partial x_i}.$$

Assuming $x_i \neq 0$, the derivative of the max-function is given by

$$\frac{\partial z_i}{\partial x_i} = \frac{\partial}{\partial x_i}\max(x_i, 0) = \begin{cases} 1 & \text{if } x_i > 0 \\ 0 & \text{else} \end{cases}.$$

Thus the backpropagation expression sought is simply

$$\frac{\partial L}{\partial x_i} = \begin{cases} \frac{\partial L}{\partial z_i} & \text{if } x_i > 0 \\ 0 & \text{else} \end{cases}.$$

Listing 3: `relu_forward`

```
function y = relu_forward(x)
    y = max(x, 0);
end
```

Listing 4: `relu_backward`

```
function dldx = relu_backward(x, dldy)
    dldx = dldy.*(x > 0);
end
```

Exercise 4

Given the loss function $L(\boldsymbol{x}, c) = -x_c + \log\left(\sum_{j=1}^{m} e^{x_j}\right)$, compute the expression for $\frac{\partial L}{\partial x_i}$.

Solution

First, let $S(\boldsymbol{x}) = \sum_{j=1}^{m} e^{x_j}$, then the loss can be written as $L(\boldsymbol{x}, c) = -x_c + \log(S(\boldsymbol{x}))$. Using the chain rule and $\frac{\partial S(\boldsymbol{x})}{\partial x_i} = e^{x_j}$ the derivative can be computed.

$$\frac{\partial L}{\partial x_i} = -\frac{\partial x_c}{\partial x_i} + \frac{\partial S(\boldsymbol{x})}{\partial x_i}\frac{\partial \log(S(\boldsymbol{x}))}{\partial S} = -\delta_{ic} + e^{x_i}\frac{1}{S(\boldsymbol{x})} = z_i - \delta_{ic}$$

where

$$z_i = \frac{e^{x_i}}{S(\boldsymbol{x})} \quad \text{and} \quad \delta_{ic} = \begin{cases} 1 & \text{if } i = c \\ 0 & \text{otherwise} \end{cases}$$

Listing 5: `softmax_forward`

```
...
    i = vec(double(labels)) + vec((0:batch-1)*features);
    L = (sum(-x(i)) + sum(log(sum(exp(x), 1))))/batches;
end
```

Listing 6: `softmax_backward`

```
...
    ex = exp(x);
    z = ex./sum(ex, 1);

    i = vec(double(labels)) + vec((0:batch-1)*features);
    delta_ic = zeros(features, batch);
    delta_ic(i) = 1;
    dldx = (z - delta_ic)/batch;
end
```

Exercise 5

Implement gradient descent with momentum.

Solution

In gradient descent with momentum the parameters are updated according to the rule below

$$\boldsymbol{m}_n = \mu \boldsymbol{m}_{n-1} + (1-\mu)\frac{\partial L}{\partial \boldsymbol{w}}$$
$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \alpha \boldsymbol{m}_n$$

If weight decay is added the momentum computation is the same, but the update rule is changed slightly

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \alpha \left(\boldsymbol{m}_n + \lambda \boldsymbol{w}_n\right),$$

where λ is the weight decay parameter.

Listing 7: training excerpt

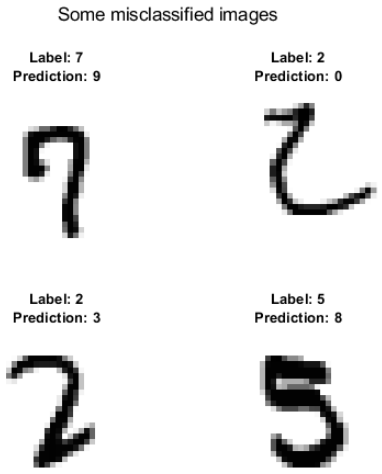
```
...
    % Computing the momentum
    mu = opts.momentum;
    momentum{i}.(s) = mu*momentum{i}.(s) + ...
        (1 - mu) * grads{i}.(s);

    % Updating weights
    net.layers{i}.params.(s) = net.layers{i}.params.(s) - ...
        opts.learning_rate * (momentum{i}.(s) + ...
        opts.weight_decay * net.layers{i}.params.(s));
...
```

Exercise 6

The filters from the first convolution layer as well as some misclassified images are shown below.

(a) We can clearly see some shapes in the filters which correspond to arcs of a circle, as well as horizontal and vertical lines.



(b) Clearly the classifier is good at recognizing patterns, and the misclassified images closely resemble the prediction.

The classifier has picked up on characteristic shapes in the images, and uses these to classify images. We can see clearly that the classifier uses small pieces of the images and not the whole image to determine the correct class. A human would instantly see that the top right image is not a zero, as the circle is not complete. The network does however not consider this rule in the evaluation, and hence misclassifies this image of the number two as a zero.

Table 1: Confusion matrix and precision and recall values for the trained network evaluated on a test set .

	Label									
Prediction	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
0	972	0	6	0	2	2	9	0	2	2
1	0	1112	1	0	0	0	3	1	0	4
2	0	0	898	0	0	0	0	6	0	0
3	0	4	34	989	0	8	1	5	0	2
4	0	1	4	0	939	0	3	0	0	3
5	0	1	0	1	0	828	1	0	0	1
6	0	0	1	0	1	2	918	0	0	0
7	1	1	18	3	1	1	0	995	3	7
8	4	16	69	12	3	41	23	6	959	4
9	3	0	1	5	36	10	0	15	10	986
PPV	0.992	0.980	0.870	0.979	0.956	0.928	0.958	0.968	0.985	0.977
TPR	0.977	0.992	0.993	0.948	0.988	0.995	0.996	0.966	0.843	0.925

The confusion matrix gives more nuance to the performance of the NN. Take for example the digit 8. The low recall rate (TPR) of 0.843 indicates that if an image of the number 8 is shown, the network has trouble distinguishing it from other numbers, while the high precision rate (PPV) of 0.985 shows that if the network predics an image as an 8, the network is often correct. On the

other hand, take the number 3. The network has very high recall rate meaning that if the network is shown a 3 it often predicts correctly, while it often incorrectly predicts other numbers as a 3.

Analysing the confusion matrix gives a deeper understanding of the network. Both the precision and recall rate must be studied to draw any conclusions of the performance.

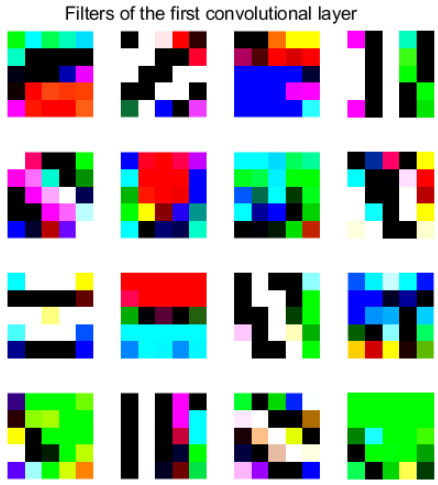
Table 2: Number of parameters used in each layer for this network.

Layer	Type	# Parameters
1	Input	0
2	Conv	416
3	ReLu	0
4	Maxpool	0
5	Conv	6416
6	ReLu	0
7	Maxpool	0
8	FC	7850
9	Softmax	0
Total:		14682

The bulk of the parameters is in the fully connected layer and in the second convolution layer, only a fraction are in the first convolution layer. This indicates that not a lot of logic is needed to detect the small patterns in the digits, instead a lot of logic is needed to piece the small pieces together and determine which digit is presented.

Exercise 7

I tried a few things to increase the performance of the network. One thing that didn't work was haphazardly adding layers everywhere. The best result, presented below, were performed by a network mimicking AlexNet. My network starts with 2 convolution networks, each followed by a ReLU layer and max pooling, next there are 3 consecutive convolution layers and the network is rounded off with two fully connected layers. The network was trained for about 70 000 iterations using all available training data. See the table at the bottom for details. Some filters from the first convolution layer as well as some misclassified images are shown below.



(a) There are some contours, but the colors seem very random.

(b) The network has a very hard time with this set. From this subsample

Table 3: Confusion matrix and precision and recall values for the trained network evaluated on a test set.

	Label									
Prediction	airplane	auto-mobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	766	146	112	38	55	18	23	36	141	164
automobile	2	492	1	1	0	0	1	0	1	15
bird	28	8	376	26	50	27	35	26	7	11
cat	92	107	270	694	283	393	259	204	58	157
deer	12	20	98	64	534	45	126	125	8	17
dog	3	13	80	94	35	452	21	71	5	18
frog	9	18	26	35	9	24	511	7	7	23
horse	8	18	13	22	15	21	9	520	1	50
ship	75	133	23	25	18	20	15	10	771	144
truck	5	45	1	1	1	0	0	1	1	401
PPV	0.766	0.492	0.376	0.694	0.534	0.452	0.511	0.520	0.771	0.401
TPR	0.511	0.959	0.633	0.276	0.509	0.571	0.764	0.768	0.625	0.879

This data set is clearly harder to classify than the mnist set, even though they have the same number of labels. The problem with this set is that there are very small details in the images that separates one image from another. This is completely different from the digit recognition problem, since digits were created to be easy to distinguish from each other, whereas a truck and an automobile or

a deer and a horse are very similar. This is reflected in the PPV and TPR rates. For example, the network has a TPR rate of nearly 90% for trucks, meaning if the network predicts an image as a truck it is almost 90% likely a truck. On the contrary, if the network is shown an image of a truck it only correctly classifies this as a truck about 40% of the time. The confusion matrix is a very good too for analyzing the performance of the network, and the training- or validation accuracy do not reflect the performance.

Overall this network is not very good, and probably needs some work. Perhaps using a deeper network with more advanced techniques such as dropout can increase performance. Also, studying AlexNet more closely could shed light on the reasoning for the construction of the network, and help give insight to how to tune filter sizes and other parameters for my poor reconstruction of their network.

Table 4: Number of parameters used in each layer for my network.

Layer	Type	# Parameters
1	Input	0
2	Conv	2 432
3	ReLu	0
4	Maxpool	0
5	Conv	51 264
6	ReLu	0
7	Maxpool	0
8	Conv	102 464
9	Conv	102 464
10	Conv	76 848
11	ReLu	0
12	Maxpool	0
13	FC	590 592
14	FC	7690
15	Softmax	0
Total:		933 754

Similar to the network used for the mnist data set the bulk of the parameters are in the fully connected layers.