UNIVERSITY OF OSLO

Faculty of mathematics and natural sciences

Exam in:	IN 5400/IN 9400 — Machine Learning for Image Analysis
Day of examination:	June 4 2021
Examination hours:	9.00 – 13.00
This exercise set consists of 17 pages.	
Appendices:	None
Permitted aids:	All

Information and tasks are given on the next pages.

(Continued on page 2.)

It is important that you read the following two (2) cover pages carefully before you start.

General information:

- Read the entire exercise text before you start solving the exercises. Please check that the exam paper is complete.
- Important messages during the exam are given directly from the course teacher on the course's semester page. It is therefore important that you check the course's semester page regularly.
- Your answer should reflect your own independent work and should be a result of your own learning and work effort.
- All sources of information are allowed for written home exams. If you reproduce a text from books, online articles, etc., a reference to these sources must be provided to avoid suspicions of plagiarism. This also applies if a text is translated from other languages.
- You are responsible for ensuring that your exam answers are not available to others during the exam period, neither physically nor digitally.
- Remember that your exam answers must be anonymous; do not state either your name or that of fellow students.
- If you want to withdraw from the exam, press the menu at the top right of Inspera and select "Withdraw".

Collaboration during the exam:

It is not allowed to collaborate or communicate with others during the exam. Cooperation and communication will be considered as attempted cheating. A plagiarism control is performed on all submitted exams where text similarities between answers are checked. If you use notes that have been prepared in collaboration with others before the exam, this might be detected in a plagiarism control. Such text similarities will be considered an attempt at cheating.

Cheating / Dishonesty / Unfair advantage during an examination

Read about what is considered cheating on UiO's website. `https://www.uio.no/english/studies/examinations/cheating/index.html`

Further notes:

(Continued on page 3.)

- File upload: You have been given 30 min. extra time for uploading files (e.g. digital hand drawings). Please see also `https://www.uio.no/english/studies/examinations/submissions/options-for-hand-drawings.html`.
- You can mix handwriting, hand drawings, digital drawings and digitally typed documents. Handwritten drawings are okay. Handwritten text on drawings is okay. Digitally typed text is preferred over handwritten text because of less potential for misreading.
- You are responsible for the readability of your handwriting. You are responsible for the checking of the quality of scans of any drawings or handwritten parts. Remember that you can scan also using mobile phones.
- Calculations: show your intermediate steps of calculations.
- Most of your answers should include a discussion, typically a few sentences.
- **submit one pdf-file in inspera**. In this file, the exercises should be answered in given order. Any figures or drawings MUST be included at the correct place in the document.

  Windows: you can import scanned drawings into some office document format with typed text and export as pdf. Linux: same as for windows. Also: pdfunite allows to concatenate many pdfs into one.
- If you lack information in the exam text or think that some information is missing, you may make your own assumptions, as long as they are justifiable within the context of the exercise. In such a case, you should make it clear what assumptions you have made.
- Plan your time so that you can try to answer as many subtasks as possible.

Contact information:

User support for the exam `https://www.mn.uio.no/english/studies/exam/user-support.html`

(Continued on page 4.)

Exercise 1

1a

What is the advantage of using residual connections in a CNN for training? Name two.

- They improve the gradient flow when computing the gradients. Reason is that the gradient flows back along the shortcut via an identity operator because of

$$h(x) = x + C_1 \circ C_2(x)$$
$$\frac{\partial h}{\partial x_d} = 1 + \frac{\partial C_1 \circ C_2}{\partial x_d}(x)$$

- They allow to unlearn easily learnt pattern detectors in the convolutions parallel to the shortcut, because the parameters in C_1 and C_2 can be shrunk to zeros during gradient descent. Then $h(x) = x$ is nothing more than the identity. Furthermore it results in a initialisation of the network such that it is approximately very shallow because the blocks with residual connections are approximately identity mappings.

1b

Consider the following code:

```
class subnet(nn.Module):
  ...

  def forward(self,x):

    x = self.convblock17(x)
    x = self.convblock3(x)
    x = self.strangeblock1(x)
    x = self.convblock11(x)
    x = self.convblock9(x)
    x = self.strangeblock2(x)
    x = self.convblock81(x)

    return x
```

How do you have to change the code such that there is a residual connection from the input to self.strangeblock1 until the output of self.convblock9 ?

(Continued on page 5.)

```
class subnet(nn.Module):
  ...

  def forward(self,x):

    x = self.convblock17(x)
    xs = self.convblock3(x)

    x = self.strangeblock1(xs)
    x = self.convblock11(x)
    x = xs+ self.convblock9(x)
    x = self.strangeblock2(x)
    x = self.convblock81(x)

    return x
```

Exercise 2

2a

If you start training with a too high learning rate such as SGD with learning rate on the order of 1.0 or larger, what can go wrong and how can you notice that something goes wrong ?

- a too high learning rate can lead to divergence of learnt parameters. That is $\|w_t\| \to \infty$. This can be seen by looking at the train loss. If the train loss and the validation loss are both not going down as expected, compared to the case when training a model with a lower learning rate.

2b

How can you diagnose overfitting to the training set during training? Please make a sketch if you would look at a plot for doing so.

- when the training loss goes further down, but the validation data loss does not go down at the same rate.
- sketch omitted here

2c

Given a partition of a dataset into train, validation, and test part, you are considering two options:

(option 1) select the model from the best epoch on the validation set, select the best learning rate on the test set

(Continued on page 6.)

(option 2) select both, the best epoch and the best learning rate on the validation set

What difference can occur between (option 1) and (option 2)?

- if one selects hyperparameters on the test set, then one could start to overfit to specifics of the data in the testset. This means that the choice of hyperparameters gives a higher performance on the particular testset and a lower performance on other datasets drawn from the same source as the test set used.

Exercise 3

You are given with a fixed dataset of images, a fixed neural network artchitecture for classification where the weights are initialised using a model trained on another dataset, and a dataloader which resizes images to the desired size, subtracts the mean, divides by standard deviation and provides them as input to the neural network. The learning rate is set to 0.001.

3a

Given that you cannot change the neural network architecture, finetuning initialization or the batchsize, you cannot collect more data, nor can you change the train/validation/test splits, name three ways which you can do to improve the prediction performance on the validation and test set and justify why they likely will help to improve performance more than other possible options.

- learning rate decay scheme
- data augmentation at training time
- ensemble models

Exercise 4

4a

Explain one similarity and one difference between a fully-connected layer and a 1D(one dimensional)-convolution layer.

- similarity: both compute a linear function over a part of the input (both have trainable parameters)

(Continued on page 7.)

- difference: the linear layer computes a linear function over the whole input, while a 1-d convolution uses as input a window of size kernel size. A linear layer applies different trainable parameters to compute each output value, while a 1d-convolution layer applies the same trainable parameters to compute the output values of one channel.

4b

Explain two differences between batchnormalization and adaptive instance normalization.

- instance normalization normalizes each instance separately, in that sense it is as if the batch size is fixed to 1
- the scale and the bias parameter are directly trainable parameters in batchnorm. in instance norm they are inputs from other network modules, and thus not directly trainable

Exercise 5

5a

In a 2-dimensional convolution layer, you are using a kernel of size $(7, 5)$ and stride 3 with padding 4. If your input feature map has a spatial shape of $(69, 90)$, what will be the spatial shape of the output feature map? Note that we do not mention here the number of channels.

Show your intermediate computations and the result.

- formula:

$$O = \text{floor}\left(1 + \frac{I + 2pad - ksize}{s}\right)$$

$$O = \text{floor}\left(1 + \frac{69 + 2 * 4 - 7}{3}\right) \qquad = \text{floor}(1 + \frac{70}{3}) = 24$$

$$O = \text{floor}\left(1 + \frac{90 + 2 * 4 - 5}{3}\right) \qquad = \text{floor}(1 + \frac{93}{3}) = 32$$

5b

A 1-dimensional convolutional layer has input layer l_0 and uses a kernel of size 3 and stride 2. The output feature map will be named layer l_1. You use – on top of this layer l_1 – another convolutional layer with kernel of size 5 and stride 3 . The output feature map will be named layer l_2. You use – on top of this layer l_2 – another convolutional layer with kernel of

(Continued on page 8.)

size 3 and stride 1. The output feature map will be named layer l_3.

How many spatial elements of the original input l_0 will be used as input for computing the output of a single neuron of layers l_2 and l_3?

You need to answer this only for neurons in the middle of the feature map. You can ignore padding effects at the boundaries of the feature map. Show your calculations.

You can also use drawings, but show your calculations in any case.

- $$R_k = R_{k-1} + (F_k - 1) \prod_{i=1}^{k-1} S_i$$
$$F_1 = 3, S_1 = 2$$
$$F_2 = 5, S_2 = 3$$
$$F_3 = 3, S_3 = 1$$
$$R_1 = 1 + (3-1) * 1 = 3$$
$$R_2 = 3 + (5-1) * 2 = 11$$
$$R_3 = 11 + (3-1) * 2 * 3 = 23$$

Exercise 6

6a

In what case does the surrogate-based attack have advantages compared to a Carlini-Wagner or an Iterative Gradient sign attack ?

- the surrogate attack does not make use of gradients of the original model. it can be used on models which either have no well-defined gradients, or models for which one has no access to the internals to compute gradients efficiently

6b

What problem can occur when one performs an adversarial attack using gradients and constant learning rate on the output of a tanh layer of a neural network which is used to produce regression outputs? The alternative would be to use the inputs to the tanh layer.

- the tanh function saturates at both ends. in these regions its gradients become very small. in these regions one either progresses very slowly or one would need to choose a very high constant learning rate, but with a very high learning rate one would make very large steps once the inputs to the tanh are closer to zero, and the gradient optimization can become unstable then

(Continued on page 9.)

Exercise 7

Refer to the neural network in figure 1. The equation for any neuron $k \in \{a, b, c, \ldots\}$ is

$$n(bias + \sum_k w_k k + \sum_l v_l x_l), \tag{1}$$

where $n(\cdot)$ is some activation function, and $v_l = 0$ for neurons which have no direct inputs x_l. $w_k = 0$ for neurons which have no inputs from other neurons.

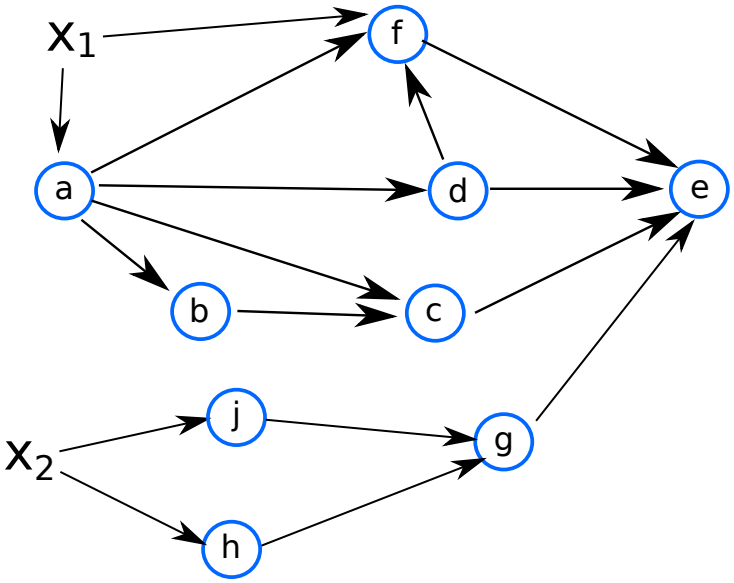


Figure 1: Mini Neural Network

Note: the following points of this task are advice. Write the expression in terms of (if needed)

- $\frac{\partial e}{\partial k}$, where k is a neuron directly connected to the loss function e
- $\frac{\partial k_1}{\partial k_2}$, where k_2 is direct input to k_1
- and $\frac{\partial k}{\partial x_l}$ if x_l is input to the neural network (otherwise use $\frac{\partial k_1}{\partial k_2}$)

(advice) You do **not need** to plugin how $\frac{\partial k_1}{\partial k_2}$ or $\frac{\partial k}{\partial x_l}$ looks like.

(advice) You **do not need** to multiply out terms in parentheses, so $(a + b)c$ or $((a + b)c + (d + e)f)g$ is fine to keep it like that!

Write down an expression for the following gradients of the neural network in figure 1:

(Continued on page 10.)

7a

$\frac{\partial e}{\partial x_2}$

- ...

$$\begin{aligned}\frac{\partial e}{\partial x_2} = \frac{de}{dx_2} &= \frac{\partial e}{\partial g}\frac{dg}{dx_2} = \frac{\partial e}{\partial g}(\frac{\partial g}{\partial j}\frac{dj}{dx_2} + \frac{\partial g}{\partial h}\frac{dh}{dx_2}) \\ &= \frac{\partial e}{\partial g}(\frac{\partial g}{\partial j}\frac{\partial j}{\partial x_2} + \frac{\partial g}{\partial h}\frac{\partial h}{\partial x_2})\end{aligned}$$

7b

$\frac{\partial e}{\partial x_1}$

- ...

$$\begin{aligned}\frac{\partial e}{\partial x_1} = \frac{de}{dx_1} &= \frac{\partial e}{\partial f}\frac{df}{dx_1} + \frac{\partial e}{\partial d}\frac{dd}{dx_1} + \frac{\partial e}{\partial c}\frac{dc}{dx_1} \\ &= \frac{\partial e}{\partial f}(\frac{\partial f}{\partial x_1} + \frac{\partial f}{\partial a}\frac{da}{dx_1} + \frac{\partial f}{\partial d}\frac{dd}{dx_1}) + \frac{\partial e}{\partial d}\frac{dd}{dx_1} + \frac{\partial e}{\partial c}\frac{dc}{dx_1} \\ &= \frac{\partial e}{\partial f}(\frac{\partial f}{\partial x_1} + \frac{\partial f}{\partial a}\frac{\partial a}{\partial x_1} + \frac{\partial f}{\partial d}\frac{dd}{dx_1}) + \frac{\partial e}{\partial d}\frac{dd}{dx_1} + \frac{\partial e}{\partial c}\frac{dc}{dx_1} \\ &= \frac{\partial e}{\partial f}(\frac{\partial f}{\partial x_1} + \frac{\partial f}{\partial a}\frac{\partial a}{\partial x_1} + \frac{\partial f}{\partial d}\frac{\partial d}{\partial a}\frac{da}{dx_1}) + \frac{\partial e}{\partial d}\frac{\partial d}{\partial a}\frac{da}{dx_1} + \frac{\partial e}{\partial c}\frac{dc}{dx_1} \\ &= \frac{\partial e}{\partial f}(\frac{\partial f}{\partial x_1} + \frac{\partial f}{\partial a}\frac{\partial a}{\partial x_1} + \frac{\partial f}{\partial d}\frac{\partial d}{\partial a}\frac{\partial a}{\partial x_1}) + \frac{\partial e}{\partial d}\frac{\partial d}{\partial a}\frac{\partial a}{\partial x_1} + \frac{\partial e}{\partial c}\frac{dc}{dx_1} \\ &= \frac{\partial e}{\partial f}(\frac{\partial f}{\partial x_1} + \frac{\partial f}{\partial a}\frac{\partial a}{\partial x_1} + \frac{\partial f}{\partial d}\frac{\partial d}{\partial a}\frac{\partial a}{\partial x_1}) + \frac{\partial e}{\partial d}\frac{\partial d}{\partial a}\frac{\partial a}{\partial x_1} + \frac{\partial e}{\partial c}(\frac{\partial c}{\partial b}\frac{\partial b}{\partial a} + \frac{\partial c}{\partial a})\frac{da}{dx_1} \\ &= \frac{\partial e}{\partial f}(\frac{\partial f}{\partial x_1} + \frac{\partial f}{\partial a}\frac{\partial a}{\partial x_1} + \frac{\partial f}{\partial d}\frac{\partial d}{\partial a}\frac{\partial a}{\partial x_1}) + \frac{\partial e}{\partial d}\frac{\partial d}{\partial a}\frac{\partial a}{\partial x_1} + \frac{\partial e}{\partial c}(\frac{\partial c}{\partial b}\frac{\partial b}{\partial a} + \frac{\partial c}{\partial a})\frac{\partial a}{\partial x_1}\end{aligned}$$

Exercise 8

Consider the following table which encodes a directed graph, which defines connections in a neural network. An entry of 1 in row i and column k denotes that neuron k is a function of input i, or graphically $i \longrightarrow k$. F is the output neuron.

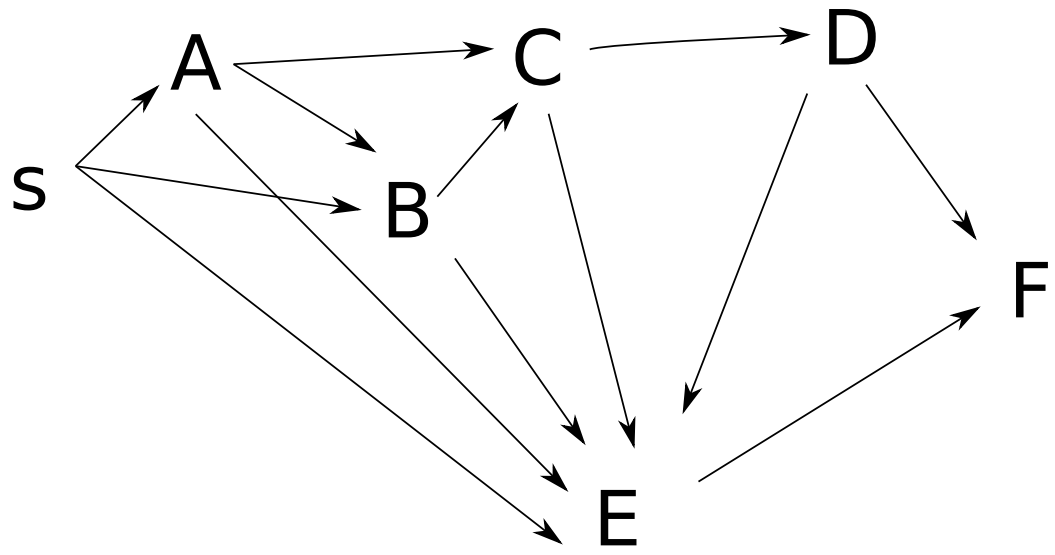
	S	A	B	C	D	E	F
S	0	1	1	0	0	1	0
A	0	0	1	1	0	1	0
B	0	0	0	1	0	1	0
C	0	0	0	0	1	1	0
D	0	0	0	0	0	1	1
E	0	0	0	0	0	0	1

(Continued on page 11.)

8a

Draw the corresponding neural net

- ...



8b

What neural network class would be defined if the diagonal part of this table would have entries of 1? State in max 3 sentences why you think so. The diagonal part are entries in the table for $[A, A], [B, B], ...$

- in that case neurons have connections to itself. This is the case for cells in a recurrent neural network with hidden states. A cell can be called repeatedly with new inputs and its hidden state from a previous call.

Exercise 9

9a

What can you use in a Generator of a GAN to increase the feature map size during the forward pass? Name 2 architecture choices and possible values of parameters relevant to increase the feature map size.

- fractionally strided convolution (deconvolution) with a fractional stride larger than 1
- upsampling layers such as bilinear upsampling and an increased output size.

(Continued on page 12.)

9b

What is the role of a discriminator model in a GAN? Describe briefly two approaches how discriminators can be can be defined. Describe refers here to firstly, naming these approaches, and secondly describing the outline of these approaches, in particularly, what they take as input and output, and what they are optimizing for.

- a discriminator could be used to learn to discriminate between samples from the dataset and samples which are the output of a generatore
- a discriminator could be used to learn the Lipschitz-1 function which is used to compute the wasserstein distance between the distribution of the real data and the distribution of the generated data.

9c

Give one example of a formula for a loss for training the generator. Explain briefly how it ensures that a generator learns to produce desired output by discussing the meaning of having a very low or minimal loss for your particular loss formula.

- $$L_G = -\frac{1}{n}\sum_{i=1}^{n}\log(D(G(x_i))),\ x_i \sim P$$

  for a discriminator which classifies real versus fake and for which $D(x) = 1$ has the meaning of a sample being perceived as real.

  If $L_G = 0$, then it implies that $D(G(x_i)) = 1\ \forall x_i$ and thus every generated samples appears to be indistinguishable from a real sample

Exercise 10

Assume that you start out with an already-trained convolutional neural net (CNN) for image classification. You are then set with the task of detecting and localizing objects of interest in images. There are three types of objects that you want to detect in the images; birds, balloons and baboons.

To start with, let us assume that there is at most one such object of interest in each image.

One approach to handling this is to modify the network so that it produces 8 output numbers, apply a suitable loss function, get appropriate training data, and then (re-)train this new network.

(Continued on page 13.)

10a

In such a case, explain/define what this array of 8 output-numbers would be (how to interpret such an array of network outputs), and suggest an appropriate loss function.

- 4 class probabilities (includes a background class) + 4 bounding-box coordinates. A conditional two-termed loss; e.g. a cross-entropy loss for the classification part, and e.g. a L1 or L2 loss for the bounding-box (if $\neq$ background).

10b

Now lets assume that images can have multiple objects of interest. With an object detection network which is capable of detecting of multiple objects of interest, we often end up having thousands of possible output bounding boxes for each input image. Many of them will thus likely detect the same object.

How is this typically resolved?

- The non-max suppression algorithm as a post-processing step.

Exercise 11

Consider this piece of code.

```
class mypredictionnet(nn.Module):
  def __init__(self):
    super(mypredictionnet,self).__init__()
    self.block1 = nn.Conv2d(3, out_channels= 8, kernel_size=5)
    self.block2 = nn.Conv2d(8, out_channels= 8, kernel_size=5)
    self.block3 = nn.Conv2d(8, out_channels= 32, kernel_size=3)
    self.block4 = nn.Conv2d(32, out_channels= 32, kernel_size=3)
    self.block5 = nn.Conv2d(32, out_channels= 32, kernel_size=3)
    self.block6 = nn.Conv2d(32, out_channels= 32, kernel_size=3)
    self.somepooling = torch.nn.AdaptiveAvgPool2d(output_size = 1 )
    self.fc = nn.Linear(32,10)


  def forward(self,x):
    x= torch.nn.functional.relu( self.block1(x) )
    x= torch.nn.functional.relu( self.block2(x) )
    x= torch.nn.functional.relu( self.block3(x) )
    x= torch.nn.functional.relu( self.block4(x) )
    x= torch.nn.functional.relu( self.block5(x) )
    x= torch.nn.functional.relu( self.block6(x) )
```

(Continued on page 14.)

```
    x= self.somepooling(x)
    x = torch.flatten(x, 1)

    x = self.fc( x )

    return x

def trainsome():

  useddevice= torch.device('cuda:0')

  dataloader = torch.utils.data.DataLoader(#here some working code)
  lossfunction =  #here some working loss function is defined
  net = mypredictionnet().to(useddevice)

  trainables=[]
  for name,params in net.named_parameters():
    if 'block4' in name:
      trainables.append(params)
    elif 'block5' in name:
      trainables.append(params)
    elif 'block6' in name:
      trainables.append(params)

  optimizer = torch.optim.SGD(trainables, lr=0.001, momentum=0.9)

  net.train(True)
  for data in dataloader:
    # data['img'] will be the features, data['labels'] will be the labels
    inputs = data['img'].to(useddevice)
    labels = data['labels'].to(useddevice)

    optimizer.zero_grad()
    outputs = net(inputs)
    trainloss = lossfunction (outputs , labels)
    trainloss.backward()
    optimizer.step()
```

11a

Why this model will perform poorly on more complex problems going beyond the simplest datasets like MNIST? You will see a poor performance on a test data set. Note: there is no issue with the size of the learning rate or kernel sizes. You also do not need to check for missing ":" at the end of for loops and other small syntactical noise.

- block1 to block3 and also the fc layer are never trained during the

(Continued on page 15.)

optimization. Note that all parameters are randomly initialized.

11b

How would you fix that? Briefly explain two different ways to fix the problem and why each of the two different ways will make the model learn better provided that the data has some information in it.

- variant 1: Therefore, include block1 to block3 and fc in the optimizer.
- variant 2: initialize the whole net with weights pretrained from another task, then freezing lower parts likely wont harm the training much, and include the layer named fc in the optimizer.

Exercise 12

Something more involved about Wasserstein distance. It is possible to get a top grade even when making really big mistakes in this task.

Remember the dual form of the Wasserstein distance, estimated on two sets of samples as

$$\sup_{f \in Lip_1} \frac{1}{|A|} \sum_{x_i \in A} f(x_i) - \frac{1}{|B|} \sum_{x_i \in B} f(x_i)$$

Furthermore: K is a Lipschitz-constant of f if:

$$\forall x_1, x_2 : \ |f(x_1) - f(x_2)| \leq K \|x_1 - x_2\|$$

Lip_1 is the space of all functions with Lipschitz constants smaller or equal to 1:

$$f \in Lip_1 \Leftrightarrow \forall x, y : \ \|f(x) - f(y)\| < \|x - y\|$$

Suppose you have two samples, one each from two different distributions:

$$(3, 1) \text{ from distribution A}$$
$$(-2, 2) \text{ from distribution B}$$

consider the function $f_{\mathbf{w}}(x_1, x_2) = w_1 x_1 + w_2 x_2$ such that $w_1 = 0.3$, $w_2 = -0.5$

Complete the following two steps to show that f does not compute the Wasserstein distance between the estimates of the two distributions represented by their only available samples.

(Continued on page 16.)

12a

Show firstly that $f \in Lip_1$ with a Lipschitz constant $K(w) < 1$
Then show secondly that for the function

$$h(w) = \frac{1}{|A|} \sum_{x_i \in A} f_w(x_i) - \frac{1}{|B|} \sum_{x_i \in B} f_w(x_i)$$
$$w = (0.3, -0.5)$$

you have $\nabla h(w) \neq 0$.

- $$|f_{\mathbf{w}}(x_1, x_2) - f_{\mathbf{w}}(y_1, y_2)| = |w \cdot (x - y)| \leq \|w\| \|x - y\|$$
$$\|w\| = \sqrt{0.09 + 0.25} < \sqrt{0.36} = 0.6 < 1$$
$$K(w) = \sqrt{0.34}$$

- $$h(w) = w \cdot ((3, 1) - (-2, 2)) = w \cdot (5, -1)$$
$$\nabla h(w) = (5, -1)$$

(Outside of your task:) Why this works ? This implies, that you can update the current weight in $f = f_w$ using:

$$w_2 = w + \epsilon \nabla h(w)$$

and you would obtain for a sufficiently small $\epsilon > 0$: $h(w_2) > h(w) = f_w$. Then you have done all what is necessary to prove that f does not compute the Wasserstein distance between the estimates of the two distributions. Below I will explain why are these two properties sufficient to prove:

It shows, firstly, that there exists a function with larger value for $g(w_2)$, which, however, is not said to be in Lip_1.

However, since $f \in Lip_1$ with a Lipschitz constant $K(w) < 1$, a sufficiently small change in its parameter w would still result in a Lipschitz constant $K(w_2) < 1$ as well, and thus $g(w_2) \in Lip_1$. For $\epsilon > 0$ sufficiently small, we can realize a sufficiently small change in its parameter w.

Exercise 13 PhD students only

Answers from MSc students will be ignored :)

(Continued on page 17.)

13a

Consider the inner product between two vectors:

$$s(v, w) = \frac{v \cdot w}{\|v\|\|w\|} = \frac{\sum_{d=1}^{D} v_d w_d}{\|v\|\|w\|}$$

Why is the normalized inner product considered a similarity measure? Discuss this by giving three pairs of vectors v, w as examples.

- if $v = \lambda w$ for $\lambda > 0$, then $s(v, w) = 1$,
- if v is orthogonal to w, means these vectors have a 90 degree angle, then $s(v, w) = 0$,
- if $v = \lambda w$ for $\lambda < 0$, means these vectors point into opposite directions , then $s(v, w) = -1$,

13b

If you take the VGG-16 architecture, and you would train it on a dataset of fixed size:

– the VGG16 with one fully-connected layer removed

– the VGG16 with one fully-connected layer added at the top.

For which of these two choices would you expect more or less overfitting as compared to the original VGG-16 architecture for suitable choice of training hyperparameters? Justify your answer in at most 3 sentences.

- removing would likely lead to less overfitting as one has for the same data much less parameters. Adding a fully-connected layer would in many cases lead to more overfitting because of more parameters and same training data.