VIR 2022 **Name:** ____________________

Midterm test

Variant: A Points ____________________

1. **ML regression**: You have a sonar sensor mounted in the front of a submarine. The sensor is based on sending the sound impulse and receiving the signal reflected from a potential obstacle in front of the submarine. You can measure the time-of-flight, i.e. the time it takes from sending till receiving reflected signal, therefore the distance to the obstacle is roughly proportional to the half of the time-of-flight and the speed of sound propagation in the water. In addition to that a static time delay is also present, therefore you decide to **model** this relation as the affine function:

$$y = w_1 x + w_0$$

- 1.1 Let us assume that y values were estimated by a device whose outputs are damaged by additive zero-mean Laplace **noise**:

$$\text{Laplace}(\mu, b) = \frac{1}{2b}\exp(-\frac{|y-\mu|}{b}),$$

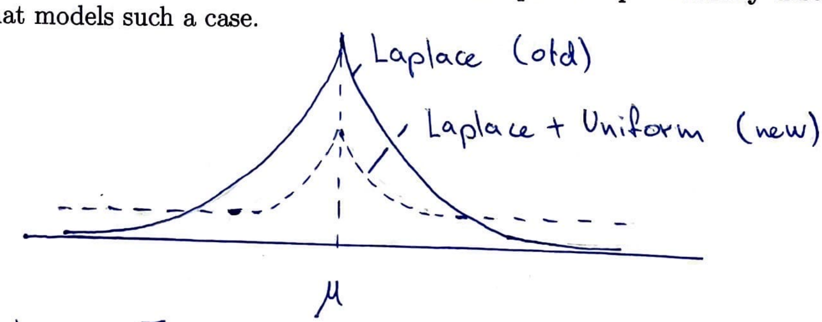
where μ is its mean value and $b \in \mathbb{R}^+$ is its diversity (quantity proportional to its variance). You are given a training set $\mathcal{D} = \{(x_1, y_1) \dots (x_N, y_N)\}$ of measured time-of-flights x_i and corresponding obstacle distances y_i. Define the probability distribution $p(y|x, \mathbf{w})$ of predicted values $y \in \mathbb{R}$ given the measurement $x \in \mathbb{R}$ and weights $\mathbf{w} \in \mathbb{R}^2$. Write down the optimization problem, which corresponds to the maximum likelihood estimate of the model parameters $\mathbf{w}$ and simplify the resulting optimization problem if possible to provide a **loss function**.

$$p(y|x,\vec{w}) = \frac{1}{2b} e^{-\frac{|w_1x+w_0-\mu|}{b}}$$

MLE: $\underset{\vec{w}}{\text{argmax}} \prod p(y_i|x,\vec{w})$

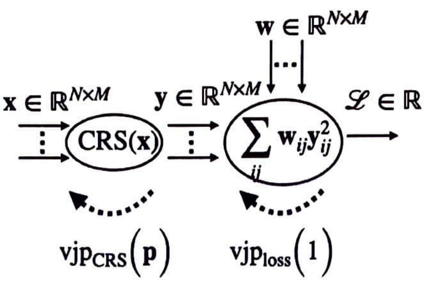
Loss: $\frac{1}{N}\sum -\log[p(y_i|x_i,\vec{w})]$

- ⋆ 1.2 Consider now, that time-to-time an unidentified floating object (UFO) such as fish or garbage appeared between the sonar and the obstacle, when y-value has been determined, therefore some small fraction of measurements y_i rather corresponds to the distance from the UFO. **Draw the shape of a probability distribution** that models such a case.



Lecture 5, slide 31

2. **Vector-jacobian-product:** Consider computational graph below:



The graph consists of two functions:

a function that perform Cyclic Row Shift (CRS) of the input matrix $\mathbf{x} \in \mathbb{R}^{N\times M}$ and returns matrix $\mathbf{y} \in \mathbb{R}^{N\times M}$, that is shifted in the row direction down and the last row is moved to the first row. For example if $N = M = 3$, the function works as follows:

$$\mathbf{y} = \text{CRS}(\mathbf{x}) = \text{CRS}\left(\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}\right) = \begin{bmatrix} x_{31} & x_{32} & x_{33} \\ x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{bmatrix}$$

b the loss function that is defined as weighted sum of squares:

$$\mathcal{L} = \sum_i \mathbf{w}_i \mathbf{y}_i^2$$

If the problem seems too complicated, solve a simpler variant (with -1 point penalty), where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$ are N-dimensional vectors only.

- 2.1 Derive gradient of the loss $\mathcal{L}$ wrt $\mathbf{x}$ (i.e. $N \times M$ matrix).

$$\frac{\partial \mathcal{L}}{\partial x_{11}} = \frac{\partial \mathcal{L}}{\partial y_{21}} \frac{\partial y_{21}}{\partial x_{11}} \quad \frac{\partial \mathcal{L}}{\partial y_{21}} = 2w_{21}y_{21} \quad \frac{\partial y_{21}}{\partial x_{11}} = 1 \quad \frac{\partial \mathcal{L}}{\partial x_{11}} = 2w_{21}y_{21} \overset{y_{21}=x_{11}}{=} 2w_{21}x_{11}$$

$$\frac{\partial \mathcal{L}}{\partial \vec{x}} = \begin{bmatrix} 2w_{21}x_{11} & 2w_{22}x_{12} & 2w_{23}x_{13} \\ 2w_{31}x_{21} & 2w_{32}x_{22} & 2w_{33}x_{23} \\ 2w_{11}x_{31} & 2w_{12}x_{32} & 2w_{13}x_{33} \end{bmatrix}$$

- 2.2 Define vector-jacobian-product function $\text{vjp}_{\text{CRS}}(\mathbf{p})$, where $\mathbf{p} \in \mathbb{R}^{\mathrm{N\times M}}$ is assumed to be the upstream gradient vector. Discuss computational and memory requirements on the backward pass using Jacobian multiplications and vjp_{CRS} function (one short sentence only).

ICRS ... inverse cyclic row shift

↳ shift in the other direction than CRS

$\text{vjp}_{CRS}(\vec{x}) = \text{ICRS}(\vec{x})$

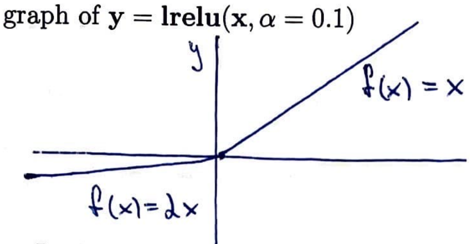
Computational and memory requirements of vjp_{CRS} are much smaller than Jacobian multiplications. As we do not need to compute anything we just shift the values of upstream gradient.

3. **Sigmoid regression:** Consider sigmoid regression problem, where the sigmoid function $\sigma(w_2x^2 + w_1x + w_0)$, that is parameterized by $\mathbf{w} \in \mathbb{R}^3$, is fitted into (x,y)-points in the least squares sense. Circle correct answers and strike-through the incorrect answers:

- 3.1 If x_i comes from uniform distribution $U(-1, 1)$, $y_i \in R^+$ and $w_0, w_2 >> 0$ are very large positive numbers, the gradient is
  - small
  - normal
  - large
  - negative
  - positive
  - undefined.
- 3.2 The criterion function that is minimized is
  - non-decreasing
  - non-increasing
  - always positive
  - always negative
  - monotonously increasing
  - differentiable
  - convex
  - concave
  - always smaller than $\sum_i y_i^2$.
- 3.3 The problem of fitting the function $y = \sigma(w_2x^2 + w_1x + w_0)$ in the least squares sense has closed-form solution.
  - TRUE
  - FALSE

4. **Leaky-ReLU:** We define Leaky Rectified Linear Unit function with parameter α as maximum of two linear functions: $\mathbf{y} = \mathbf{lrelu}(\mathbf{x}, \alpha) = \max\{\mathbf{x}, \alpha\mathbf{x}\}$. The function maps single input $\mathbf{x}$ on single output value $\mathbf{y}$, i.e. $\mathbf{y}, \mathbf{x} \in \mathbb{R}$. The parameter $\alpha \in \mathbb{R}$ corresponds to its slope for negative inputs.

- 4.1 Draw graph of $\mathbf{y} = \mathbf{lrelu}(\mathbf{x}, \alpha = 0.1)$



- 4.2 Define a Leaky Rectified Linear Unit $\mathbf{lrelu}(\mathbf{x}, \alpha)$ in pseudocode.

```
def lrelu (x, α):
    if x ≥ 0:
        return x
    else:
        return αx
```

- 4.3 Define the gradient of the $\mathbf{lrelu}(\mathbf{x}, \alpha)$ activation function in pseudocode. The function has a single argument $\mathbf{x}$ and outputs $\frac{\partial \mathbf{lrelu}(\mathbf{x}, \alpha)}{\partial \mathbf{x}}$. Hint: Break up the function into two separate cases (if-else).

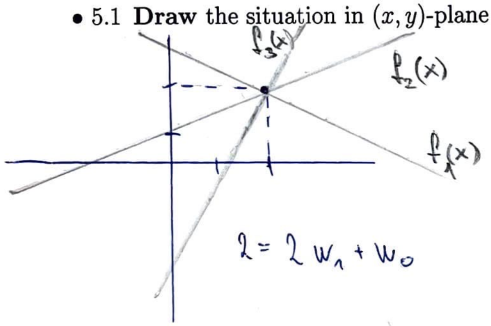
```
def lrelu_grad(x, α):
    if x ≥ 0
        return 1
    else:
        return α
```

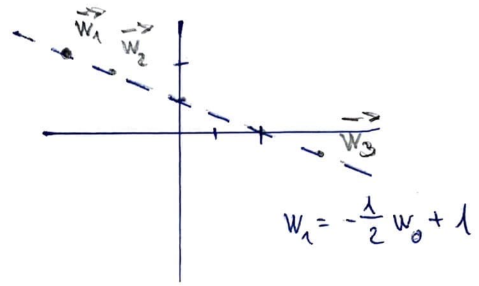
- 4.4 Let us assume that you set hyper parameter $\alpha = 0$. What happens in GD, when learning reaches the state in which all training samples cause negative input into this lrelu-layer?

gradient = $\vec{0}$ => weights stop updating

5. **Prior:** Consider problem of fitting the line $y = w_1 x + w_0$ in the least squares sense, where training set consists of a single training example: $\mathcal{D} = \{(x_1 = 2, y_1 = 2)\}$. Since there is more than one hypothesis consistent with this training set, you decide to use a regularization.

- 5.1 **Draw** the situation in (x, y)-plane and (w_0, w_1)-plane.





- 5.2 **Suggest a suitable regularization** and write down the underlying optimization problem that you have to solve.

regularization: $r(\vec{w}) = w_1^2 + w_2^2 = \|\vec{w}\|^2$

optimization p.: $\min\limits_{\vec{w}} \left(\| y - w_1 x + w_0 \|^2 + \|\vec{w}\|^2 \right)$

$\min\limits_{\vec{w}} \left(g(x, y, \vec{w}) \right)$

- 5.3 What is the globally **optimal solution** $\mathbf{w}^*$ under the suggested regularization?

$$g(x, y, \vec{w}) = y^2 - 2y(w_1 x + w_0) + w_1^2 x^2 + 2 w_0 w_1 x + w_0^2 + w_1^2 + w_0^2$$

$$\frac{\partial g}{\partial w_0} = -2y + 2w_1 x + 4w_0 \xrightarrow{x=y=2} 4w_0 + 4w_1 - 4$$

$$\frac{\partial g}{\partial w_1} = -2xy + 2w_1 x^2 + 2w_0 x + 2w_1 \xrightarrow{x=y=2} 4w_0 + 10w_1 - 8$$

$$w_0 + w_1 - 1 = 0 \quad \Rightarrow w_0 = -w_1 + 1$$

$$\mathrm{grad}(g) = 0 \Rightarrow \quad 2w_0 + 5w_1 - 8 = 0$$

$$-2w_1 + 2 + 5w_1 - 8 = 0$$

$$3w_1 = 6 \Rightarrow w_1 = 2$$

$$w_0 = -1$$

$$\vec{w}^* = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

6. **Conv2D feedforward and backward pass:** In all questions, assume that the stride denotes length of convolutional stride, pad denotes symmetric zero-padding, rate is dilatation rate of convolution. conv stands for convolution layer.

You are given input feature map (image) $\mathbf{x}$ and kernel $\mathbf{w}$:

$$\mathbf{x} = \begin{array}{|c|c|c|}\hline 1 & 0 & 2 \\ \hline 2 & 1 & -1 \\ \hline 0 & 0 & 2 \\ \hline \end{array} \qquad \mathbf{w} = \begin{array}{|c|c|}\hline 2 & 0 \\ \hline 0 & 1 \\ \hline \end{array}$$

- 6.1 Compute output of $\mathbf{y} = \text{conv}(\mathbf{x},\ \mathbf{w},\ \text{stride} = 1,\ \text{pad} = 0,\ \text{rate} = 2) =$

$y = 4$

- 6.2 Let us assume that convolution is followed by another layer (say function $\mathbf{z} = f(\mathbf{y})$ with $\mathbf{z} \in \mathbb{R}$). What is the **dimensionality of upstream gradient for the convolutional layer**:

$\frac{\partial z}{\partial y}$

1×1

$z \in \mathbb{R}$ $\quad$ $y \in \mathbb{R}$

- 6.3 What is $\text{vjp}_{\textsf{conv}}(\mathbf{p})$ of this particular convolution with respect to weights, where $\mathbf{p}$ is upstream gradient?

$\text{vjp}_{\text{conv\_w}}(\vec{p}, \vec{x}) = \text{conv}(\vec{x}, \vec{p}, \text{stride}=2, \text{pad}=0, \text{rate}=1)$