VIR 2021 | **Name:** Chytrý Student

Exam test

Variant: A | Points 20

1. You are given batch of two one-dimensional training examples $\mathbf{B} = \{x_1 = 2, x_2 = 4\}$ that are goes through the Batch-norm layer with $\gamma = 6$, $\beta = -1$ and $\epsilon = 0$:

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

- Compute jacobian of batch norm output with respect to the learnable parameters.
Hint: Output of the batch-norm layer for this batch is two-dimensional.

$\mu_\mathcal{B} = \frac{1}{2}(2+4) = 3$

$\sigma_\mathcal{B}^2 = \frac{1}{2}\left[(2-3)^2 + (4-3)^2\right] = 1$

$\hat{x}_1 = \frac{2-3}{\sqrt{1}} = -1$

$\hat{x}_2 = \frac{4-3}{\sqrt{1}} = 1$

$y_1 = -\gamma + \beta$

$y_2 = \gamma + \beta$

$\frac{\partial y_i}{\partial \gamma} = \hat{x}_i$

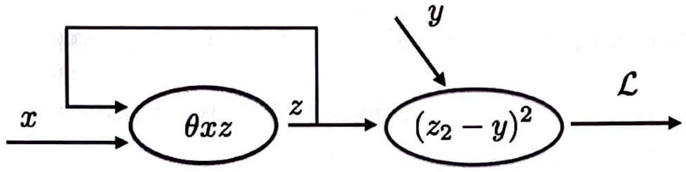
$\frac{\partial y_i}{\partial \beta} = 1$

$$J = \begin{bmatrix} \frac{\partial y_1}{\partial \beta} & \frac{\partial y_1}{\partial \gamma} \\ \frac{\partial y_2}{\partial \beta} & \frac{\partial y_2}{\partial \gamma} \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

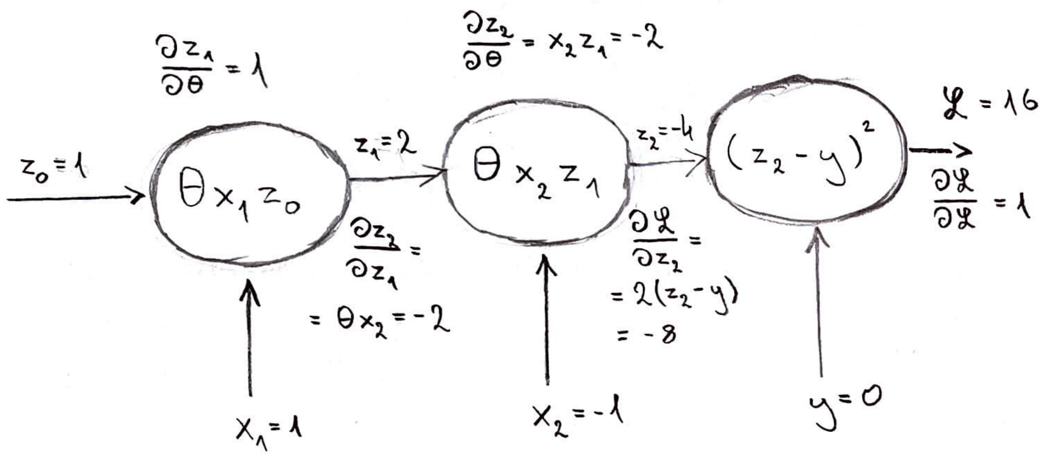
2. Consider recurrent neural network depicted on the image below. The network is initialized with parameter $\theta = 2$ and initial hidden state $z_0 = 1$. You are given the following input sequence:

time=1	time=2
$x_1 = 1$	$x_2 = -1$

The loss is computed only on the **last** hidden state z_2 as a L2 distance from $y = 0$. Estimate gradient $\frac{\partial \mathcal{L}}{\partial \theta}$ of the loss $\mathcal{L}$ with respect to θ.



Hint: Unroll the network in time, to obtain a usual feedforward network. Do the backpropagation as usual.



$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial z_2} \cdot \frac{\partial z_2}{\partial \theta} + \frac{\partial \mathcal{L}}{\partial z_2} \cdot \frac{\partial z_2}{\partial z_1} \cdot \frac{\partial z_1}{\partial \theta} = (-8) \cdot (-2) + (-8)(-2)(1) = 32$$

- Why do we keep batch size larger than 1, when using batch norm layer?

Pokud by velikost batche byla rovna 1, pak $\hat{x} = 0$ a tedy výstupem by byl pouze parametr β.

- How do you update γ and β (using jacobian) to increase the output values of the batch norm layer?
  Hint: Look at the each row of the jacobian.

γ: Snadno bychom dokázali, že $\sum_i \hat{x}_i = 0$, tedy změnou γ nezměníme výsledný součet výstupních hodnot.

β: $\frac{\partial y_i}{\partial \beta} = 1$ -> změnou β přímo úměrně zvětšujeme výstupní hodnoty

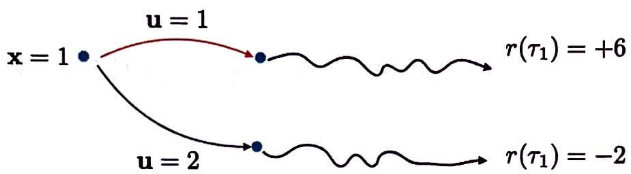
- What are the benefits of using the batch norm layer inside the neural networks?

- síť se trénuje rychleji
- nepotřebujeme malý learning rate ke konvergenci
- jednodušší inicializace vah
- působí trochu jako regularizace

3. Consider stochastic discrete policy, that selects action $\mathbf{u}$ in state $\mathbf{x}$ according to the following probability distribution:

$$\pi_\theta(\mathbf{u}|\mathbf{x}) = \begin{cases} \sigma(\theta\mathbf{x}+1) & \text{if } \mathbf{u}=1 \\ 1-\sigma(\theta\mathbf{x}+1) & \text{if } \mathbf{u}=2 \end{cases}$$

with scalar parameter $\theta = -1$. This policy maps one-dimensional state $\mathbf{x}$ on the probability distribution of two possible actions $\mathbf{u}=1$ or $\mathbf{u}=2$. Consider simplified MDP, where stochastic discrete policy can perform the action only if the system is in state $\mathbf{x}=1$ (e.g. hit the ball either by forehand or backhand), then the ball follows the trajectory τ_1 or τ_2. The resulting trajectory is evaluated by reward function $r(\tau_1) = +6$, $r(\tau_2) = -2$.



- What are values of the advantage function of this policy in state $\mathbf{x}=1$:

$A(\mathbf{u}=1, \mathbf{x}=1) = Q(u=1, x=1) - V(x=1) = 6-2 = 4$

$A(\mathbf{u}=2, \mathbf{x}=1) = Q(u=2, x=1) - V(x=1) = -2-2 = -4$

$Q(u=1, x=1) = r(\tau_1) = 6$

$Q(u=2, x=1) = r(\tau_2) = -2$

$V(x=1) = E[r] = \sigma(\theta x + 1)\cdot r(\tau_1) + [1-\sigma(\theta x+1)]\cdot r(\tau_1)] =$

$= 6\cdot\underbrace{\sigma(1-1)}_{\sigma(0)=1/2} + 2\cdot\underbrace{\sigma(1-1)}_{1/2} - 2 = 3 + 1 - 2 = 2$

- You are given training trajectory $\tau = [\mathbf{x}_1 = 1, \mathbf{u}_1 = 1, ...]$. This trajectory consists of the single transition (outlined by red color) followed by the rest of the trajectory. Policy could have decided only the action in state $\mathbf{x} = 1$. Estimate A2C policy gradient:

$$\left.\frac{\partial \log \pi_\theta(\mathbf{u}|\mathbf{x})}{\partial \theta}\right|_{\substack{\mathbf{x}=\mathbf{x}_1 \\ \mathbf{u}=\mathbf{u}_1}} \cdot A(\mathbf{u}=\mathbf{u}_1, \mathbf{x}=\mathbf{x}_1) = \frac{1}{2} \cdot 4 = 2$$

$$\left.\frac{\partial \log \pi_\theta(\mu|x)}{\partial \theta}\right|_{\substack{x_1 \\ \mu_1}} = \frac{\partial \log\left[\sigma(\theta x_1 + 1)\right]}{\partial \theta} = \frac{1}{\sigma(\theta x_1 + 1)} \cdot \sigma(\theta x_1 + 1)\left[1 - \sigma(\theta x_1 + 1)\right] x_1 =$$

$$= x_1 - x_1 \sigma(\theta x_1 + 1) = 1 - \sigma(0) = \frac{1}{2}$$

4. You are given the following function

$$f(w) = 3w^2 - 1.$$

Consider gradient descend algorithm (SGD), which updates the scalar weight $w \in \mathcal{R}$ as follows:

$$w^k = w^{k-1} - \alpha \left.\frac{\partial f^\top(w)}{\partial w}\right|_{w=w^{k-1}},$$

where α denotes its learning rate.

- For which set of α values the SGD **converges** (at least slowly)?

$\alpha_{\text{converge}} \in (0, 1/3)$

$$\frac{\partial f^\top(w)}{\partial w} = 6w$$

$$w^k = w^{k-1} - \alpha 6 w^{k-1} = (1 - 6\alpha) w^{k-1}$$

$$w^k = (1 - 6\alpha)^k w^0$$

- For which set of α values the SGD **oscillates**?

$\alpha_{\text{oscillate}} \in \{0, 1/3\}$

- For which set of α values the SGD **diverges**?

$\alpha_{\text{diverge}} \in (-\infty, 0) \cup (1/3, \infty)$

- What is the best learning rate α^*, which guarantees the **fastest convergence** rate for arbitrary weight initialization $\mathbf{w}^0$

$\alpha^* = 1/6$

$1 - 6\alpha^* = 0$

$\alpha^* = \frac{1}{6}$

- Is it possible to find such α-subsets (i.e. the subsets where the SGD converges, diverges and oscillates) for $f(w) = |w|$? If so find it, if not explain the reasons and explain how to adjust the algorithm to achieve the convergence.

$$\frac{\partial f(w)}{\partial w} = \begin{cases} 1, & \text{pro } w > 0 \\ -1, & \text{pro } w < 0 \end{cases}$$

pro $w = 0$ derivace neexistuje

SGD algoritmus bude skoro pokaždé oscilovat. Jediný případ, kdy nebude oscilovat, je takový, že si zvolíme w^0 a α takové, že "nepřekročíme" minimum (nezmění se nám derivace). To platí když $k\alpha = w^0$, $k \in \mathbb{N}$.