

Assignment 1: Problem set

20 points. You may use lecture notes and a calculator. Show the requested steps. Submit typed or clearly scanned answers as A1_Written_Answers.pdf, labeled A1.W1-A1.W4.

A1.W1. Images to predictions

[4 points]

Questions 1 and 2 use a small version of the MNIST network: 2 by 2 grayscale images, two ReLU hidden units, and two output classes. Both linear layers have biases.

(a) A batch has shape $(3, 1, 2, 2)$. Give its shape after flattening the image axes and the shape of the final logits. (1 point)

(b) How many trainable parameters does this network have? Show the count for each layer, including biases. (1 point)

For (c)-(d), use one image, flattened as a column vector. Weight rows correspond to output units.

$$x = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad W_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}, \quad b_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

$$W_2 = \begin{bmatrix} 1 & 2 \\ -1 & 1 \end{bmatrix}, \quad b_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad y = 1.$$

$$s = W_1 x + b_1, \quad h = \text{ReLU}(s), \quad z = W_2 h + b_2.$$

(c) Calculate the hidden pre-activations, hidden activations and logits. (1 point)

(d) Calculate the softmax probabilities and cross-entropy loss. The correct class has zero-based index 1. Use natural logarithms. (1 point)

A1.W2. Understanding backpropagation

[6 points]

Use the same network, example and cross-entropy loss as Question 1. Work at the given parameter values, before any update.

(a) Calculate the loss gradient with respect to the two logits. (1 point)

(b) Calculate the gradients with respect to the output weight matrix and output bias. Show the matrix product for the weight gradient. (2 points)

(c) Calculate the gradients with respect to the hidden-layer weight matrix and bias. Briefly explain why one row of the weight gradient is zero. (2 points)

(d) In PyTorch, a first call to `backward()` fills the parameter gradients above. At unchanged parameters, a student recomputes the same forward pass and loss, then calls `backward()` again without clearing gradients. What will `W2.grad` contain? Name a call that clears parameter gradients before the next minibatch. (1 point)

Problem set (continued)**A1.W3. One AdamW update****[5 points]**

A scalar parameter is updated with AdamW. Use the AdamW update from the notebook, with bias correction and separate decay of the old parameter. The supplied gradient is from the data loss only.

$$w_0 = 1, \quad m_0 = v_0 = 0, \quad g_1 = 2, \quad \eta = 0.1, \quad \beta_1 = \beta_2 = 0.5, \quad \lambda = 0.2, \quad \epsilon = 0.$$

- (a) Calculate the first and second moments after the first gradient. (1 point)
- (b) Calculate the bias-corrected moments for this step. (1 point)
- (c) Calculate the updated parameter. Show the weight-decay and adaptive-gradient terms separately. (1 point)
- (d) The next data gradient is zero. Calculate the new uncorrected moments. Is the adaptive-gradient part of this second update zero? Explain in one sentence. You do not need to calculate the second parameter value. (2 points)

A1.W4. Training and evaluation**[5 points]**

A BatchNorm layer processes one feature with learned scale 2 and shift -1. Ignore epsilon. During training, calculate the batch variance using divisor 2.

- (a) For the training batch (1, 3), calculate the mean, variance and both BatchNorm outputs. (2 points)
- (b) In evaluation mode, the stored mean is 0 and stored variance is 4. Calculate both outputs for the same batch. A student leaves the model in training mode and uses only `torch.no_grad()` for evaluation. Will this give the evaluation outputs? If not, name the missing call. (2 points)
- (c) Separately, an inverted-dropout layer receives activations (2, 4), with drop probability 0.5 and sampled keep mask (1, 0). Give its training output and its evaluation output. (1 point)