DEEP LEARNING: ASSIGNMENTS WITH SOLUTIONS

Assignment 1 (Node statistics)**.** Let us consider a neuron in a linear layer of a classification network. Its output is given by

$$y = \sum_{i=1}^{n} w_i x_i,$$

where x is the output of the preceding layer. Let us consider the statistics of x over the training data and assume that the components x_i are statistically independent and identically distributed with zero mean and variance σ^2. The weight components w_i are initialized i.i.d. with zero mean and variance $\widetilde{\sigma}^2$. Compute the mean and variance of y.

Solution. Since w_i and x_i are statistically independent, we obtain the mean of y by

$$\mathbb{E}[y] = \sum_{i=1}^{n} \mathbb{E}[w_i]\mathbb{E}[x_i] = 0. \tag{1}$$

To compute the variance of y, we use that $\mathbb{V}[XY] = \mathbb{V}[X]\mathbb{V}[Y]$ and $\mathbb{V}[X+Y] = \mathbb{V}[X] + \mathbb{V}[Y]$ hold for any pair of statistically independent random variables X and Y. We obtain

$$\mathbb{V}[y] = \sum_{i=1}^{n} \mathbb{V}[w_i]\mathbb{V}[x_i] = n\widetilde{\sigma}^2\sigma^2. \tag{2}$$

□

Assignment 2 (Backpropagation)**.**
Let $x \in \mathbb{R}^N$ be a vector with components x_i for $i = 1, \dots N$ and consider a layer performing the following computation:

$$y_i = a(x_i + x_{i+2}) + b \quad \text{for } i = 1 \dots N-2. \tag{3}$$

Given the gradient of the loss function in y, $g := \nabla_y L \in \mathbb{R}^{N-2}$, compute the gradient of the loss in a, b and x.

Solution.

$$\frac{\mathrm{d}L}{\mathrm{d}b} = \sum_{i=1}^{N-2} \frac{\mathrm{d}L}{\mathrm{d}y_i}\frac{\partial y_i}{\partial b} = \sum_{i=1}^{N-2} \frac{\partial L}{\partial y_i} = \sum_{i=1}^{N-2} g_i. \tag{4}$$

$$\frac{\mathrm{d}L}{\mathrm{d}a} = \sum_{i=1}^{N-2} \frac{\mathrm{d}L}{\mathrm{d}y_i}\frac{\partial y_i}{\partial a} = \sum_{i=1}^{N-2} g_i(x_i + x_{i+2}). \tag{5}$$

$$\frac{\mathrm{d}L}{\mathrm{d}x_j} = \sum_{i=1}^{N-2} g_i \frac{\partial y_i}{\partial x_j} = \sum_{i=1}^{N-2} g_i a\big([\![j{=}i]\!] + [\![j{=}i{+}2]\!]\big) = \begin{cases} ag_j & \text{if } j \le 2, \\ a(g_j + g_{j-2}) & \text{if } j = 2, \dots N-2, \\ ag_{j-2} & \text{if } j \ge N-2. \end{cases} \tag{6}$$

□

Assignment 3 (SGD with Regularization)**.**
Consider a regularized loss function $\tilde{L}(\theta) = L(\theta) + \frac{\lambda}{2}\|\theta\|^2$. Let θ^t be the current parameter estimate and g^t be the gradient of L at θ^t.

a) Give an update step for an SGD-like algorithm that applies a variance reduction technique to stochastic gradients g^t in order to obtain smoothed estimates $\tilde{g}_t$.

b) Solve the following proximal step problem

$$\theta^{t+1} = \arg\min_{\theta} \Big[\langle \tilde{g}^t, \theta - \theta^t \rangle + \frac{\lambda}{2}\|\theta\|^2 + \frac{1}{2\varepsilon'}\|\theta - \theta^t\|^2 \Big]. \tag{7}$$

Solution. **a)** To reduce the variance of stochastic gradients g^t we will use exponentially weighted average with parameter q.

$$\tilde{g}^t := \tilde{g}^{t-1}(1-q) + g^t q. \tag{8}$$

Then we write standard SGD step using the gradient $\tilde{g}^t + \lambda\theta^t$ — the smoothed gradient of L plus the gradient of regularization at θ^t:

$$\theta^{t+1} := \theta^t - \varepsilon(\tilde{g}^t + \lambda\theta^t). \tag{9}$$

b)

$$\theta^{t+1} = \arg\min_{\theta} \Big[\langle \tilde{g}^t, \theta - \theta^t \rangle + \frac{\lambda}{2}\|\theta\|^2 + \frac{1}{2\varepsilon'}\|\theta - \theta^t\|^2 \Big]. \tag{10}$$

Solving for stationary point:

$$0 = \frac{\mathrm{d}}{\mathrm{d}\theta} = \tilde{g}^t + \lambda\theta + \frac{1}{\varepsilon'}(\theta - \theta^t). \tag{11}$$

We find:

$$\theta^{t+1} = \frac{\theta^t - \varepsilon'\tilde{g}^t}{\varepsilon'\lambda + 1}. \tag{12}$$

□

Remark. We can check that by setting $\varepsilon' = \frac{\varepsilon}{1-\lambda\varepsilon}$ this solution matches the common SGD step (9), *i.e.* for quadratic regularization linearizing it or considering explicitly in the proximal problem is equivalent.

Assignment 4 (Adversarial attack)**.** Let us consider a neural network for classification with predictive class log probabilities given by the vector $f(x;\theta) \in \mathbb{R}^K$. An attacker wants to find a perturbed image $\tilde{x}$ satisfying $|\tilde{x}_i - x_i| < \varepsilon$ for all i such that it would minimize the probability of predicting the correct label y.

Formulate the attacker's task as an optimization problem using a *linear approximation* of f in the box $|\tilde{x}_i - x_i| < \varepsilon$. Solve this problem.

Solution. The log probability of the correct label is $f_y(x;\theta)$ and its linear approximation in the neighbourhod of x is given by

$$f_y(\tilde{x};\theta) \approx f_y(x;\theta) + g^T(\tilde{x} - x), \tag{13}$$

where g denotes the gradient $\nabla_x f_y(x;\theta)$. The attackers task is

$$g^T(\tilde{x} - x) \to \min_{\tilde{x}} \tag{14}$$

$$\text{s.t.} \quad |\tilde{x}_i - x_i| < \varepsilon \quad \forall i. \tag{15}$$

It decomposes into independent tasks for each $\tilde{x}_i$ with solution $\tilde{x}_i^* = -\varepsilon \operatorname{sign}(g_i)$. □

Assignment 5 (KL divergence and cross entropy)**.**
Assume that the training data are given by a generator $p^*(y, x)$. We want to learn the conditional distribution $p(y \mid x;\theta)$ in the form of a neural network parametrized by θ. Prove that minimizing $\mathbb{E}_{p^*(x)}[D_{\mathrm{KL}}(p^*(y \mid x) \,\|\, p(y \mid x;\theta))]$ is equivalent to minimizing the expected cross-entropy of $p(y \mid x;\theta)$ relative to $p^*(y \mid x)$, where the expectation is taken over $p^*(x)$.

Solution. Let us expand the KL divergence for a give x:

$$D_{\mathrm{KL}}(p^*(y \mid x) \,\|\, p(y \mid x;\theta)) = \int_y p^*(y \mid x) \log \frac{p^*(y \mid x)}{p(y \mid x;\theta)} \tag{16}$$

$$= \underbrace{\int_y p^*(y \mid x) p^*(y \mid x)}_{\text{does not depend on } \theta} \underbrace{- \int_y p^*(y \mid x) \log p(y \mid x;\theta)}_{\text{cross-entropy}} . \tag{17}$$

Taking expectation in $p^*(x)$ the first term still does not depend on θ and thus optimization with or without it is equivalent. □

Assignment 6 (SGD with Regularization 2)**.**
Consider a regularized loss function $\tilde{L}(\theta) = L(\theta) + \rho(\|\theta\|)$, where $\rho\colon \mathbb{R}_+ \to \mathbb{R}_+$ is a differentiable function. Let θ^t be the current parameter estimate and g be the gradient of L at θ^t. Show that the solution of the composite proximal step problem

$$\arg\min_{\theta} \Big[\langle g, \theta - \theta^t \rangle + \rho(\|\theta\|) + \frac{1}{2\varepsilon} \|\theta - \theta^t\|^2 \Big] \tag{18}$$

for a sufficiently small ε takes the form: $\theta = \frac{a}{\|a\|} l$, where $a = \theta^t - \varepsilon g$ is the usual non-regularized SGD update and and l is a root of the equation $l + \varepsilon \rho'(l) = \|a\|$.

Solution. We solve for a critical point:

$$0 = \frac{\partial}{\partial \theta} = g + \rho'(\|\theta\|)\frac{\theta}{\|\theta\|} + \frac{1}{\varepsilon}(\theta - \theta^t)$$
$$\theta(\frac{\varepsilon\rho'(\|\theta\|)}{\|\theta\|} + 1) = \theta^t - \varepsilon g. \tag{19}$$

Since $(\frac{\varepsilon\rho'(\|\theta\|)}{\|\theta\|} + 1)$ is a scalar we conclude that θ will be proportional to $\theta^t - \varepsilon g =: a$. Take the norm of the vectors on both sides in (19):

$$\|\theta\|\Big(\frac{\varepsilon\rho'(\|\theta\|)}{\|\theta\|} + 1\Big) = \|a\| \tag{20}$$
$$\varepsilon\rho'(\|\theta\|) + \|\theta\| = \|a\|.$$

Denoting $l = \|\theta\|$, we can express $\frac{\varepsilon\rho'(\|\theta\|)}{\|\theta\|} + 1 = \frac{\|a\|}{l}$. The equation (20) holds for ε sufficiently small so that the value of $\frac{\varepsilon\rho'(\|\theta\|)}{\|\theta\|}$ is positive, otherwise its absolute value needs to be taken. □

Assignment 7 (Shift of Prior)**.** A neural network with softmax activation in the last layer has been trained for classifying patterns by predicting the posterior class probabilities $p(y \,|\, x)$, $y \in K$. The relative class frequencies in the training set were $p(y)$. When applying the network, it turned out that the prior class probabilities for real data are different and equal to $p^*(y)$. Explain how to use the network as a predictor without re-training it. We assume the 0/1 loss for prediction.

Solution. Let us denote the distribution of the training data by $p(x, y)$. We have

$$p(x, y) = p(x \,|\, y)p(y) = p(y \,|\, x)p(x)$$

and the trained network estimates $p(y \,|\, x)$. Let us denote the data distribution in the application by $p_a(x, y)$. We have

$$p_a(x, y) = p(x \,|\, y)p^*(y) = p_a(y \,|\, x)p_a(x),$$

i.e. $p(x \,|\, y)$ remains unchanged and $p(y)$ changes to $p^*(y)$. Comparing the two equations we get

$$p_a(y \,|\, x) \propto \frac{p^*(y)}{p(y)}p(y \,|\, x).$$

Hence, the trained network can be used in the application just by reweighting its softmax outputs by the factors $\frac{p^*(y)}{p(y)}$ and deciding for the class with the largest reweighted output. □

Assignment 8 (K-means)**.** Let us consider the standard *k-means clustering* problem for data $x \in \mathbb{R}^n$ and K cluster centers $y_k \in \mathbb{R}^n$

$$\sum_{x \in \mathcal{T}^m} \min_k \|x - y_k\|^2 \to \min_y,$$

where $y = (y_1, \ldots, y_K)$ denotes the set of all cluster centers and $\mathcal{T}^m$ denotes the training set.

a) Propose a stochastic gradient descent method that operates in full online mode. I.e. it receives *one* example per iteration (the mini-batch size is 1). Explain why it is necessary to choose a decreasing learning rate.

b) What is the run-time complexity for a training epoch? Compare it with the run-time complexity of the standard k-means algorithm.

Solution. **a)** Given a single training example $x \in \mathcal{T}^m$, we have the objective $f(y) = \min_k \|x - y_k\|^2$ and its gradients w.r.t. the cluster centers are

$$\nabla_{y_k} f(y) = \begin{cases} 2(y_k - x) & \text{if } k = \arg\min_{k'} \|x - y_{k'}\|^2, \\ 0 & \text{otherwise.} \end{cases}$$

We obtain the following SGD algorithm for the problem.
Given a training example x do
(1) find the closest cluster center $k = \arg\min_{k'} \|x - y_{k'}\|^2$,
(2) update $y_k \to y_k + \alpha(t)(x - y_k)$,
where $\alpha(t)$ is a decreasing learning rate. The algorithm will not converge to a local minimum if the learning rate is constant. Instead, it will keep oszillating around it.

b) The run-time complexity of the SGD algorithm for one training epoch is $\mathcal{O}(nmK)$. The standard k-means algorithm iteration consists of two steps (i) assignment and (ii) update. The run-time complexity of the former dominates and is $\mathcal{O}(nmK)$. □

Assignment 9 (Backprop)**.**
Let $x \in \mathbb{R}^n$. Consider the following normalized linear layer:

$$y_i = \frac{w_i^\mathsf{T} x + b_i}{\|w_i\|},$$

where $w_i \in \mathbb{R}^n$ for $i = 1 \ldots m$, $b_i \in \mathbb{R}$ and $\|w_i\|$ is the Euclidean norm of vector w_i. Given the gradient of the loss function in y, $g := \nabla_y L \in \mathbb{R}^m$, compute gradients of the loss in w, b, x.

Solution. We will use general the total derivative rule

$$\frac{\mathrm{d}L}{\mathrm{d}\theta} = \sum_i \frac{\mathrm{d}L}{\mathrm{d}y_i} \frac{\partial y_i}{\partial \theta} = \sum_i g_i \frac{\partial y_i}{\partial \theta}. \tag{21}$$

Since y_i depends only on b_i and not on b_j for $j \neq i$ for $\nabla_b L$ we have

$$\frac{\mathrm{d}L}{\mathrm{d}b_i} = g_i \frac{\partial y_i}{\partial b_i} = \frac{g_i}{\|w_i\|}. \tag{22}$$

For $\nabla_x L$ we have

$$\frac{\mathrm{d}L}{\mathrm{d}x_j} = \sum_i g_i \frac{\partial y_i}{\partial x_j} = \sum_i g_i \frac{w_{ij}}{\|w_i\|}. \tag{23}$$

Since y_i depends only on w_i and not on w_j for $j \neq i$ for $\nabla_w L$ we have

$$\frac{\mathrm{d}L}{\mathrm{d}w_i} = \sum_i g_i \frac{\partial y_i}{\partial w_i} = \sum_i g_i \Big(\frac{x}{\|w_i\|} + (w_i^\mathsf{T} x + b_i) \frac{-w_i}{\|w_i\|^3} \Big). \tag{24}$$

□

Assignment 10 (VAE)**.**
Consider a variational autoencoder with the decoder model being a normal distribution $p(x|z) = \mathcal{N}(x; \mu(z), \sigma^2 I)$, where $x \in \mathbb{R}^d$ and σ is a parameter. Show that the optimal value of the variance σ^2 for the evidence lower bound

$$\text{ELBO} = \mathbb{E}_{p_d(x)} \mathbb{E}_{q(z|x)} \big[\log p(x|z) \big] - D_{KL}(q(z|x) \parallel p(z))$$

with the current encoder $q(z|x)$ is given by

$$\sigma^2 = \frac{1}{d} \mathbb{E}_{p_d(x)} \mathbb{E}_{q(z|x)} \big[\|x - \mu(z)\|^2 \big].$$

Solution. The density of the Normal distribution with diagonal covariance matrix is

$$p(x|z) = \Big(\frac{1}{\sqrt{2\pi}\sigma} \Big)^d \exp(-\frac{\|x - \mu(z)\|^2}{2\sigma^2}). \tag{25}$$

Respectively the log density is

$$\log p(x|z) = \log \Big(\frac{1}{\sqrt{2\pi}\sigma} \Big)^d - \frac{\|x - \mu(z)\|^2}{2\sigma^2} = -\frac{d}{2} d \log(2\pi) - d \log \sigma - \frac{\|x - \mu(z)\|^2}{2\sigma^2}. \tag{26}$$

Note that the log density is a convex function of σ. We find optimum by finding stationary points of ELBO in σ. The KL divergence term does not depend on σ and its derivative is zero. Since the expectation densities do not depend on σ, the derivative can be interchanged with expectation:

$$\frac{\partial}{\partial \sigma} \text{ELBO} = \mathbb{E}_{p_d(x)} \mathbb{E}_{q(z|x)} \big[\frac{\partial}{\partial \sigma} \log p(x|z) \big]. \tag{27}$$

We then calculate

$$\frac{\partial}{\partial \sigma} \log p(x|z) = -\frac{d}{\sigma} + \frac{\|x - \mu(z)\|^2}{\sigma^3}. \tag{28}$$

And solve

$$\mathbb{E}_{p_d(x)} \mathbb{E}_{q(z|x)} \big[-\frac{d}{\sigma} + \frac{\|x - \mu(z)\|^2}{\sigma^3} \big] = 0. \tag{29a}$$

$$\frac{d}{\sigma} = \frac{1}{\sigma^3} \mathbb{E}_{p_d(x)} \mathbb{E}_{q(z|x)} \big[\|x - \mu(z)\|^2 \big]. \tag{29b}$$

$$\sigma^2 = \frac{1}{d} \mathbb{E}_{p_d(x)} \mathbb{E}_{q(z|x)} \big[\|x - \mu(z)\|^2 \big]. \tag{29c}$$

□

Remark. The solution takes the same form as the maximum likelihood estimate of variance from supervised data samples x, z. The difference is that here we do not know the ground truth samples (x, z) and estimate them using the current encoder, *i.e.*, draw them from the distribution $p_d(x)q(z|x)$.

Assignment 11 (Mirror Descent)**.**
Solve the proximal step problem:

$$\min_x \langle \nabla f(x^0), x - x^0 \rangle + \frac{1}{\varepsilon} D(x, x^0),$$

where $x^0 \in (0, 1)$ and

$$D(x, x^0) = \sum_i (x_i \log \frac{x_i}{x_i^0} + (1 - x_i) \log \frac{1 - x_i}{1 - x_i^0}).$$

Hint: The problem is convex and can be solved by stationary point conditions.

Solution. The objective is a sum of terms where each summand i depends on x_i only. Therefore minimization decouples into independent minimizations over x_i:

$$\min_{x_i} \langle g_i, x_i - x_i^0 \rangle + \frac{1}{\varepsilon} (x_i \log \frac{x_i}{x_i^0} + (1 - x_i) \log \frac{1 - x_i}{1 - x_i^0}),$$

where $g = \nabla f(x^0)$. We solve for the critical point x_i:

$$\begin{aligned}
0 = \frac{\partial}{\partial x_i} &= g_i + \frac{1}{\varepsilon} (\log \frac{x_i}{x_i^0} - \log \frac{1 - x_i}{1 - x_i^0}) \\
0 &= -\varepsilon g_i + \log \frac{x_i}{1 - x_i} - \log \frac{x_i^0}{1 - x_i^0} \\
\log \frac{x_i}{1 - x_i} &= \log \frac{x_i^0}{1 - x_i^0} - \varepsilon g_i \\
x_i &= \text{sigmoid}\Big(\log \frac{x_i^0}{1 - x_i^0} - \varepsilon g_i \Big).
\end{aligned}$$

□

Remark. Suppose we solve these proximal problems iteratively and x^t is the current iteration. Denote $y^t = \text{logit}(x^t) = \log \frac{x^t}{1 - x^t}$, then $x^t = \text{sigmoid}(y^t)$ and on the next iteration we do not need to calculate $\log \frac{x^t}{1 - x^t}$, we could just reuse y^t. Then the iterates can be simplified to

$$\begin{aligned}
y^{t+1} &= y^t - \varepsilon g, \\
x^{t+1} &= \text{sigmoid}(y^{t+1}).
\end{aligned}$$

Assignment 12 (Attention Score Distribution)**.** Let $q, k \in \mathbb{R}^{d_K}$ be a query and a key. Assume that the entries of q and k are independent random variables with zero mean and unit variance, and that q and k are independent of each other.

a) Compute the mean and variance of the dot product $\langle q, k \rangle$.

b) Explain why dividing by $\sqrt{d_K}$ in the scaled dot-product attention is beneficial for the softmax function and stable scaling of learning when increasing d_K.

Solution. **a)** The dot product is $\langle q, k \rangle = \sum_{i=1}^{d_K} q_i k_i$. Each term $q_i k_i$ has:

$$\mathbb{E}[q_i k_i] = \mathbb{E}[q_i]\,\mathbb{E}[k_i] = 0,$$

since q_i and k_i are independent and zero-mean. By linearity:

$$\mathbb{E}[\langle q, k \rangle] = 0.$$

For the variance, since all $q_i k_i$ are mutually independent (the pairs (q_i, k_i) are independent across i):

$$\text{Var}[\langle q, k \rangle] = \sum_{i=1}^{d_K} \text{Var}[q_i k_i].$$

For independent zero-mean unit-variance random variables:

$$\text{Var}[q_i k_i] = \mathbb{E}[q_i^2 k_i^2] - (\mathbb{E}[q_i k_i])^2 = \mathbb{E}[q_i^2]\,\mathbb{E}[k_i^2] - 0 = 1 \cdot 1 = 1.$$

Therefore:

$$\boxed{\mathbb{E}[\langle q, k \rangle] = 0, \qquad \text{Var}[\langle q, k \rangle] = d_K.}$$

b) Without scaling, the logits $\langle q, k \rangle$ have standard deviation $\sqrt{d_K}$, which grows with the key dimension. For large d_K, some logits will be very large in magnitude, pushing the softmax into saturated regions where the gradient is extremely small (similar to a hard argmax). Dividing by $\sqrt{d_K}$ normalizes the logits to have unit variance regardless of d_K:

$$\text{Var}\left[\frac{\langle q, k \rangle}{\sqrt{d_K}}\right] = \frac{d_K}{d_K} = 1,$$

keeping the softmax in a well-behaved regime with meaningful gradients and ensuring stable scaling of learning when increasing d_K.

□

Assignment 13 (Attention)**.** Let the input be a sequence of n token representations, arranged as a matrix $X \in \mathbb{R}^{n \times d}$, where d is the model dimensionality. Self-attention first projects the input into queries, keys, and values:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

where:

- $W^Q \in \mathbb{R}^{d \times d_\text{K}}$, $W^K \in \mathbb{R}^{d \times d_\text{K}}$, $W^V \in \mathbb{R}^{d \times d_\text{V}}$ are learnable projection matrices,
- $Q, K \in \mathbb{R}^{n \times d_\text{K}}$ are the query and key matrices,

- $V \in \mathbb{R}^{n \times d_\text{V}}$ is the value matrix,
- We expect $d_V, d_K < d$ in practice, in particular when this is one head of a multi-head attention.

The "scaled dot-product attention" is computed as:

$$\text{Attention}(Q, K, V) = \left(\text{softmax}\left(\frac{QK^\top}{\sqrt{d_\text{K}}}\right) V\right) W^O,$$

where $W^O \in \mathbb{R}^{d_\text{V} \times d}$ is the output projection matrix.

a) Show that value projection matrix W^V can be "absorbed into" the output projection matrix W^O. What will be the size of the resulting output projection matrix $\tilde{W}^O$? Similarly, show that the key projection matrix W^K can be absorbed into the query projection matrix W^Q. Conclude that separate projections model low-rank (with rank $d_K, d_V < d$) interactions.

b) Which order of multiplication is more efficient: $((AX)W^V)W^O$ or $(A(XW^V))W^O$? Assume $n > d > d_V$.

c) Suppose we modify the self-attention mechanism by multiplying the value projections W^V on the right by an invertible matrix $R \in \mathbb{R}^{d_\text{V} \times d_\text{V}}$ and simultaneously multiplying all output projections W^O on the left by R^{-1}. Show that this transformation leaves the attention output unchanged. Show that we can also apply an orthogonal transform R simultaneously to the key and query matrices without changing the result. Conclude that the query and key projection matrices are not identifiable and only their product matters.

Solution. **a)** Let $A = \text{softmax}\big(\frac{QK^\top}{\sqrt{d_K}}\big) \in \mathbb{R}^{n \times n}$ denote the attention weights. The output is:

$$\text{Attention} = (A\,V)\,W^O = A\,(XW^V)\,W^O.$$

Absorbing W^V into W^O: Since matrix multiplication is associative, we can write $AX(W^V W^O)$. Defining $\tilde{W}^O = W^V W^O \in \mathbb{R}^{d \times d}$, we get:

$$\text{Attention} = (A\,X)\,\tilde{W}^O.$$

Thus the value projection $W^V \in \mathbb{R}^{d \times d_V}$ and output projection $W^O \in \mathbb{R}^{d_V \times d}$ are not separately identifiable — only their product $\tilde{W}^O \in \mathbb{R}^{d \times d}$ matters. The factorization through $d_V < d$ means that $\tilde{W}^O = W^V W^O$ has rank at most d_V, so the separate projections model a *low-rank* output transformation.

Absorbing W^K into W^Q: The attention logits are:

$$\frac{QK^\top}{\sqrt{d_K}} = \frac{(XW^Q)(XW^K)^\top}{\sqrt{d_K}} = \frac{X\,(W^Q {W^K}^\top)\,X^\top}{\sqrt{d_K}}.$$

Defining $\tilde{W}^Q = W^Q {W^K}^\top \in \mathbb{R}^{d \times d}$, the logits become $\frac{X\tilde{W}^Q X^\top}{\sqrt{d_K}}$. Again, $\tilde{W}^Q$ has rank at most d_K, so the separate query and key projections model a low-rank bilinear interaction between tokens.

b) Denote $A \in \mathbb{R}^{n\times n}$ the attention weight matrix. The two orderings compute the same result AXW^VW^O but differ in cost. Let us count multiplications:

•$((AX)W^V)W^O$: AX costs $n \cdot n \cdot d = n^2d$, then $(\cdot)W^V$ costs $n \cdot d \cdot d_V$, then $(\cdot)W^O$ costs $n \cdot d_V \cdot d$. Total: $n^2d + 2nd\,d_V$.

•$(A(XW^V))W^O$: XW^V costs $n \cdot d \cdot d_V$, then $A(\cdot)$ costs $n \cdot n \cdot d_V = n^2d_V$, then $(\cdot)W^O$ costs $n \cdot d_V \cdot d$. Total: $n^2d_V + 2nd\,d_V$.

Since $d > d_V$, we have $n^2d_V < n^2d$, so the second ordering $(A(XW^V))W^O$ is more efficient. The saving comes from reducing the dimension *before* the expensive $n \times n$ multiplication with the attention matrix.

c) Value–output invariance: Replace $W^V \to W^VR$ and $W^O \to R^{-1}W^O$. Then:

$$A\,X\,(W^VR)(R^{-1}W^O) = A\,X\,W^V\,(RR^{-1})\,W^O = A\,X\,W^V\,W^O.$$

The output is unchanged.

Key–query invariance: Replace $W^Q \to W^QR$ and $W^K \to W^KR$ for an orthogonal R ($R^\top = R^{-1}$). The attention logits become:

$$\frac{(XW^QR)(XW^KR)^\top}{\sqrt{d_K}} = \frac{XW^Q\,RR^\top\,{W^K}^\top X^\top}{\sqrt{d_K}} = \frac{XW^Q{W^K}^\top X^\top}{\sqrt{d_K}},$$

which is identical to the original. Therefore the attention weights and the full output are unchanged. This shows that W^Q and W^K are not individually identifiable — only their product $W^Q{W^K}^\top$ determines the attention.

□

Assignment 14 (GD for Attention)**.** Let $Q \in \mathbb{R}^{d\times r}$ be the query projection matrix and $K \in \mathbb{R}^{d\times r}$ be the key projection matrix with $r < d$. The loss $\mathcal{L}$ depends only on the product matrix $W = QK^\top$. Let G_Q be the derivative of the loss in Q and G_K be the derivative of the loss in K.

a) Express G_Q and G_K in terms of the derivative of the loss in W, denoted as G_W (not explicitly available, but will be helpful for the analysis).

b) Write GD in (Q, K) and express the resulting update $\Delta W = (Q+\Delta Q)(K+\Delta K)^\top - QK^\top$ in terms of G_W. Show that the update in W is a sum of two terms, one involving $G_WKK^\top$ and another involving $QQ^\top G_W$. Compare with GD in W if W was modelled as a free parameter. Conclude that poorly conditioned matrices Q, K may significantly slow down convergence.
Hint: the term $\Delta Q\Delta K^\top$ is of order $\mathcal{O}(\eta^2)$ and can be ignored for small learning rates η.

Solution. **a)** Since $W = QK^\mathsf{T} \in \mathbb{R}^{d\times d}$ and $\mathcal{L}$ depends on Q, K only through W, by the chain rule:

$$(G_Q)_{ij} = \frac{\partial\mathcal{L}}{\partial Q_{ij}} = \sum_{kl}\frac{\partial\mathcal{L}}{\partial W_{kl}}\frac{\partial W_{kl}}{\partial Q_{ij}} = \sum_l (G_W)_{il}K_{lj}, \tag{30}$$

since $W_{kl} = \sum_m Q_{km} K_{lm}$ and $\frac{\partial W_{kl}}{\partial Q_{ij}} = \delta_{ki} K_{lj}$. In matrix form:

$$G_Q = G_W K, \qquad G_K = G_W^\mathsf{T} Q. \tag{31}$$

b) The gradient descent updates in Q and K are:

$$\Delta Q = -\eta G_Q = -\eta G_W K, \qquad \Delta K = -\eta G_K = -\eta G_W^\mathsf{T} Q. \tag{32}$$

The resulting update in $W = QK^\top$ is (dropping $\mathcal{O}(\eta^2)$ terms):

$$\Delta W = (Q + \Delta Q)(K + \Delta K)^\mathsf{T} - QK^\mathsf{T} \tag{33}$$

$$= \Delta Q\, K^\mathsf{T} + Q\, \Delta K^\mathsf{T} + \underbrace{\Delta Q\, \Delta K^\mathsf{T}}_{\mathcal{O}(\eta^2)} \tag{34}$$

$$\approx -\eta\, G_W K K^\mathsf{T} - \eta\, Q Q^\mathsf{T} G_W. \tag{35}$$

If W were modelled as a free $d \times d$ parameter, GD would give $\Delta W = -\eta\, G_W$. Comparing, we see that the factored parameterization introduces implicit preconditioning:

$$\Delta W_{\text{factored}} = -\eta\, (G_W K K^\mathsf{T} + Q Q^\mathsf{T} G_W) \neq -\eta\, G_W = \Delta W_{\text{free}}. \tag{36}$$

The matrices KK^T and QQ^T act as preconditioners. If Q and K have singular values $\sigma_1, \ldots, \sigma_r$, the effective step size along the i-th singular direction is scaled by σ_i^2. Thus, directions corresponding to small singular values receive vanishingly small updates, while directions corresponding to large singular values are over-amplified. Poor conditioning of Q or K (large ratio $\sigma_{\max}/\sigma_{\min}$) can therefore significantly slow down convergence.

□

Assignment 15 (Key-Value Separation)**.** Consider attention with keys k_t, queries q_t, and values v_t for token t.

a) Construct an equivalent attention mechanism with keys k'_t, queries q'_t, and values v'_t such that $k'_t = v'_t$ for all t, by stacking the original keys and values into one vector. How should the query and the output projection be modified to recover the same attention output?

b) If we allow unconstrained key and output projections in the new mechanism, conclude that the new mechanism is strictly more expressive than the original one, while requiring the same storage of representations (KV cache).

c) What are the advantages and disadvantages of these two variants? Consider computation cost, memory traffic cost (often dominates the inference in LLMs), split of dimensions between keys and values (often they are of different dimensions by design).

Solution. **a)** The attention output for query q is:

$$\text{out}(q) = \left(\sum_t a_t\, v_t \right) W^O, \qquad a_t = \frac{\exp(q^\top k_t)}{\sum_{t'} \exp(q^\top k_{t'})}.$$

Stack the keys and values into a single vector:

$$k'_t = v'_t = \begin{pmatrix} k_t \\ v_t \end{pmatrix} \in \mathbb{R}^{d_K + d_V}.$$

Define augmented queries with zeros in the value part:

$$q'_t = \begin{pmatrix} q_t \\ 0_{d_V} \end{pmatrix} \in \mathbb{R}^{d_K + d_V}.$$

Then $q'^{\top} k'_t = q^{\top} k_t$, so the attention weights are unchanged. The aggregated vector is:

$$\sum_t a_t \, v'_t = \sum_t a_t \begin{pmatrix} k_t \\ v_t \end{pmatrix}.$$

To recover the original output, define a modified output projection $\tilde{W}^O = [0 \mid W^O] \in \mathbb{R}^{(d_K + d_V) \times d}$ that selects only the last d_V components:

$$\left(\sum_t a_t \, v'_t \right) \tilde{W}^O = \left(\sum_t a_t \, v_t \right) W^O = \text{out}(q).$$

Thus the original attention is exactly reproduced.

b) In the new mechanism $k'_t = v'_t \in \mathbb{R}^{d_K + d_V}$, the query $q' \in \mathbb{R}^{d_K + d_V}$ and output projection $\tilde{W}^O \in \mathbb{R}^{(d_K + d_V) \times d}$ are unconstrained. The query can now have a non-zero lower block, allowing the attention weights to depend on both k_t *and* v_t simultaneously — something the original mechanism cannot express. Moreover, the output projection can mix both the key and value parts of v'_t in arbitrary ways. Since the original mechanism is a special case (zero lower block in q', restricted $\tilde{W}^O$), the new mechanism is *strictly more expressive*. At the same time, the KV cache stores the same vectors $k'_t = v'_t$, so there is no additional memory cost compared to storing k_t and v_t separately (both require $d_K + d_V$ floats per token).

c) The tied model uses twice more arithmetic for the attention scores and aggregation (dimension $d_K + d_V$ instead of d_K and d_V separately). However, during autoregressive inference the bottleneck is *reading* the KV cache from memory, not arithmetic. The cache size is the same in both models ($d_K + d_V$ per token), so inference speed is essentially unchanged.

Meanwhile, the tied model is strictly more general: it can learn to use an arbitrary subspace of $\mathbb{R}^{d_K + d_V}$ for routing and its complement for retrieval, choosing the dimension split adaptively, or use the whole mixed representation both for attention scores and as the aggregated output. In contrast, the separated model fixes this split at architecture design time. This suggests that tied key-value representations are the better default, and the traditional separation is mainly a historical convention.

□

Assignment 16 (GD Reparameterization)**.**
Let a neural network be parameterized by a weight vector w and let $\mathcal{L}(w)$ be the loss function. Consider a re-parameterization of the model with $\tilde{w} = sw$, where $s > 0$. We initialize with $\tilde{w}^0 = sw^0$ and optimize the reparameterized loss function $\tilde{L}(\tilde{w}) := \mathcal{L}(\tilde{w}/s)$.

a) Express the gradient $\tilde{g}^0 = \nabla_{\tilde{w}}\tilde{\mathcal{L}}(\tilde{w}^0)$ (gradient of $\tilde{\mathcal{L}}$ w.r.t. $\tilde{w}$ at the point $\tilde{w}^0$) through $g^0 = \nabla_w \mathcal{L}(w^0)$.

b) Write GD step for $\tilde{L}$ in $\tilde{w}$ with learning rate α. Show that it is equivalent to GD for $\mathcal{L}(w)$ in w with a modified learning rate.

c) What would change if the network was invariant to the scale of w, i.e. $\mathcal{L}(\tilde{w}/s) = \mathcal{L}(\tilde{w})$?

Solution. **a)**

$$\nabla_{\tilde{w}}\tilde{\mathcal{L}}(\tilde{w}) = \nabla_{\tilde{w}}\mathcal{L}(\tilde{w}/s) = \nabla_{\tilde{w}}\mathcal{L}(w) = \frac{d}{dw}\mathcal{L}(w)\frac{dw}{d\tilde{w}} = \nabla_w \mathcal{L}(w)\frac{1}{s}. \tag{37}$$

Since $\tilde{w}^0$ and w^0 are equivalent points, $\tilde{g}^0 = \frac{1}{s}g^0$.

b) The resulting equivalent update of w is:

$$w^{t+1} = w^t - \alpha\frac{1}{s^2}\nabla_w \mathcal{L}(w^t). \tag{38}$$

c) The answers to the questions a,b do not change. E.g. for the gradient:

$$\nabla_{\tilde{w}}\tilde{\mathcal{L}}(\tilde{w}) = \nabla_{\tilde{w}}\mathcal{L}(\tilde{w}/s) = \frac{d}{dw}\mathcal{L}(w)\frac{dw}{d\tilde{w}} = \nabla_w \mathcal{L}(w)\frac{1}{s}, \tag{39}$$

the factor $\frac{1}{s}$ comes from $\frac{dw}{d\tilde{w}}$ in both cases.

□

Assignment 17 (Sample statistics)**.** For each input image x, the network outputs a feature map $f(x)$ of the size $C \times H \times W$. Explain (by equations / algorithm) how to compute the mean and standard deviation of feature maps over the spatial dimensions H, W as well as over all instances x in the full training dataset $\mathcal{T}$. Can you make the solution **on-line**: when it is not allowed to process more than one image at a time or to store more than several tensors of the size of the feature map or make more than one pass over the dataset.

Solution. Basic variant:

$$\mu = \frac{1}{NHW}\sum_{x\in\mathcal{T}}\sum_{i,j} f(x)_{i,j} \tag{40}$$

$$v = \frac{1}{NHW}\sum_{x\in\mathcal{T}}\sum_{i,j} (f(x)_{i,j} - \mu)^2 \tag{41}$$

$$\sigma = \sqrt{v}. \tag{42}$$

Online variant: the sum for μ can be computed incrementally, but for the variance v it does not work because each summand depends on μ, which we do not have yet. The online solution can compute the second moment

$$M = \frac{1}{NHW} \sum_x \sum_{i,j} f(x)_{i,j}^2 \tag{43}$$

and then express $v = M - \mu^2$. □

Assignment 18 (SGDW)**.** Consider regularized loss

$$\mathcal{L}^{\text{reg}}(\theta) = \mathcal{L}(\theta) + \frac{\lambda}{2}\|\theta\|_2^2. \tag{44}$$

Let $\hat{g}_t$ be the stochastic gradient of $\mathcal{L}$ at θ_t.

a) Formulate and solve proximal step problem that finds a step at θ_t by minimizing the linearized loss $\mathcal{L}^{\text{reg}}(\theta)$ at θ_t plus the proximity penalty $\frac{1}{2\alpha}(\theta - \theta_t)^2$.

b) This time, linearize only $\mathcal{L}(\theta)$, while keeping the regularizer exact, and solve the corresponding proximal step problem.

Solution. **a)** The step proximal problem in this cases is:

$$\min_\theta \langle \nabla \mathcal{L}^{\text{reg}}(\theta^t), \theta - \theta^t \rangle + \frac{1}{2\alpha}(\theta - \theta_t)^2, \tag{45}$$

where we can expand $\nabla \mathcal{L}^{\text{reg}}(\theta^t) = \nabla \mathcal{L}(\theta^t) + \lambda \theta^t$. Stationary point condition:

$$\nabla \mathcal{L}(\theta^t) + \lambda \theta^t + \frac{1}{\alpha}(\theta - \theta_t) = 0 \tag{46}$$

$$\theta = \theta^t - \alpha(\nabla \mathcal{L}(\theta^t) + \lambda \theta^t) = (1 - \alpha\lambda)\theta^t - \alpha \nabla \mathcal{L}(\theta^t). \tag{47}$$

b) The step proximal problem in this cases is:

$$\min_\theta \langle \nabla \mathcal{L}(\theta^t), \theta - \theta^t \rangle + \frac{\lambda}{2}\|\theta\| + \frac{1}{2\alpha}(\theta - \theta_t)^2. \tag{48}$$

Stationary point condition:

$$\nabla \mathcal{L}(\theta^t) + \lambda \theta + \frac{1}{\alpha}(\theta - \theta_t) = 0 \tag{49}$$

$$\alpha \nabla \mathcal{L}(\theta^t) + (\lambda\alpha + 1)\theta - \theta_t = 0 \tag{50}$$

$$\theta = \frac{1}{\lambda\alpha + 1}\big(\theta_t - \alpha \nabla \mathcal{L}(\theta^t)\big). \tag{51}$$

□

Assignment 19 (Distillation)**.** Suppose we have a labelled dataset $(x_i, y_i)_{i=1}^N$, where $y_i \in \mathcal{C}$ – finite set of class labels. We want to train a predictive model $p(y \mid x; \theta)$.

a) Define the negative log-likelihood training criterion, $\mathrm{NLL}(\theta)$.

b) In the approach, commonly known as knowledge distillation, a (smaller) student network is trained to mimic the already trained (big) teacher network. Given a teacher network with predictive probabilities $q(y \mid x)$, the student network $p(y \mid x; \theta)$ is trained to minimizing $\mathbb{E}_{x \sim \mathrm{data}} D_{KL}(q(y \mid x) \parallel p(y \mid x; \theta))$. Consider combining the distillation loss and the negative log-likelihood of the labeled data:

$$\mathcal{L}(\theta) = \sum_i D_{KL}(q(y \mid x_i) \parallel p(y \mid x_i; \theta)) + \lambda \mathrm{NLL}(\theta), \tag{52}$$

where $\lambda > 0$ is a hyperparameter. Show that this can be written as knowledge distillation (KL divergence criterion) with a modified teacher's label distribution.

Solution. **a)** $\mathrm{NLL}(\theta) = -\sum_i \log p(y_i \mid x_i)$

b)

$$D_{KL}(q(y \mid x_i) \parallel p(y \mid x_i; \theta)) = \sum_y q(y \mid x_i) \log \frac{q(y \mid x_i)}{p(y \mid x_i; \theta)} \tag{53}$$

$$= \sum_y q(y \mid x_i) \log q(y \mid x_i) - \sum_y q(y \mid x_i) \log p(y \mid x_i; \theta). \tag{54}$$

Only the second part depends on θ and is relevant for the learning. We can then combine

$$-\sum_y q(y \mid x_i) \log p(y \mid x_i; \theta) - \lambda \log p(y_i \mid x_i) = \tag{55}$$

$$-\sum_y (q(y \mid x_i) + \lambda [\![y{=}y_i]\!]) \log p(y \mid x_i; \theta) = -\sum_y \tilde{q}(y \mid x_i) \log p(y \mid x_i; \theta), \tag{56}$$

where $\tilde{q}(y \mid x_i) = q(y \mid x_i) + \lambda [\![y{=}y_i]\!]$ is the modified teacher's distribution: we add one-hot encoded true label with weight λ to the original teacher (could be re-normalized if we need to make it a distribution and express the objective as KL divergence).

□