DEEP LEARNING (SS2022)
SEMINAR 4

Assignment 1 (Weight initialization for ReLU networks)**.** When weights are initialized with a fixed standard deviation (e.g., 0.01), deep models (e.g., >8 conv layers) have difficulties to converge. In this assignment we derive a proper weight initialization for ReLU networks, following He et al. (2015). For networks with ReLU activations the linearity assumption (Glorot and Bengion's analysis in the lecture) does not hold. The goal is to achieve that variance of preactivations stay close to standardized in a deep network so that input is not magnified (up or down) exponentially in the number of layers K.

We will assume that the components of all vectors are statistically independent and identically distributed for tractability of analysis (weights are indeed independent i.i.d. at initialization).

a) Let us consider a single neuron with weight vector w and input vector x. Its pre-activation is $a = w^T x$. Let us assume

$$\mathbb{E}[x_i] = \mu,\ \mathbb{E}[x_i^2] = \chi,\ \mathbb{E}[w_i] = 0, \text{ and } \mathbb{V}[w_i] = v.$$

Prove that $\mathbb{E}[a] = 0$ and $\mathbb{V}[a] = nv\chi$, where n is the dimension of the vectors x and w.

b) Show that the distribution of a is symmetric about zero if so is the distribution of w.

c) Consider the neuron output $y = g(a)$, where g denotes the ReLU function. Conclude that $\mathbb{E}[y^2] = \frac{1}{2}\mathbb{V}[a]$.

d) Let us denote $\mathbb{V}[a] = \alpha$ and consider a ReLU network with layers $k = 1, \ldots, K$ with n_k units in each. Collect the previous steps to show the following recursive relation for the variance of pre-activations in the layer k:

$$\alpha_k = \frac{1}{2} n_{k-1} v_k \alpha_{k-1}.$$

Obtain the initialization of He et al. (2015): initialize the weights with zero mean and variance

$$\mathbb{V}[w_{ij}^k] = \frac{2}{n_{k-1}}.$$

Assignment 2 (Batch Normalization)**.** Consider a single coordinate of a linear layer given by $a = w^\mathsf{T} x + b$, where w is the weight vector, b is a scalar bias. Batch normalization after this layer takes the form:

$$y = \frac{a - \mu_\mathcal{B}}{\sigma_\mathcal{B}} \beta + \gamma, \tag{1}$$

where $\mu_\mathcal{B}$ and $\sigma_\mathcal{B}$ denote the sample mean and standard deviation of a taken over the batch:

$$\mu_\mathcal{B} = \frac{1}{n} \sum_{i=1}^{n} a_i; \qquad \sigma_\mathcal{B}^2 = \frac{1}{n} \sum_{i=1}^{n} (a_i - \mu_\mathcal{B})^2, \tag{2}$$

where a_i are the layer outputs for the i'th instance in the batch.

a) Show that batch-normalized output can be explicitly written as

$$y = \frac{w^\mathsf{T}(x - \bar{x})}{\|w\|_X}\beta + \gamma, \tag{3}$$

where $\bar{x}$ is the sample mean of the batch data, X is the batch covariance matrix: $X = \frac{1}{n}\sum_i (x_i - \bar{x})(x_i - \bar{x})^\mathsf{T}$ and $\|w\|_X$ is the Mahalanobis norm: $\|w\|_X = (w^\mathsf{T} X w)^{\frac{1}{2}}$. Cf. *layer normalization* which does not subtract mean and uses plain $\|w\|$.

Conclude that the output of batch normalization does not depend on the bias b and also does not change when the weight vector w is scaled by a positive constant.

b) What is the mini-batch sample mean and standard deviation of the BN-normalized layer, if we initialize $\beta = 1$, $\gamma = 0$? Assume, we decided to apply BN after each linear layer. Has the weight initialization from Assignment 1 still an effect for the forward pass?

c) Consider a network without BN. Let $\mu_\mathcal{B}$ and $\sigma_\mathcal{B}$ be the statistics of layer output $a = Wx + b$. We want to introduce a BN layer at this place so that it does not change the network predictions. How shall we initialize β and γ?

Assignment 3 (SGD + L2)**.** Consider a regularized loss function $\tilde{L}(\theta) = L(\theta) + \frac{\lambda}{2}\|\theta\|^2$. Let g be a stochastic gradient estimate of L (original loss) at θ. Notice that the regularization part of the objective, $\frac{\lambda}{2}\|\theta\|^2$, is known in a closed form and so its gradient g_r is non-stochastic.

- Design an SGD algorithm that applies momentum (exponentially weighted averaging) to g only but not to g_r.
- Is it equivalent to an SGD with the momentum applied to both g and g_r, possibly with a different settings of λ, momentum and learning rate?

Assignment 4 (Mixup)**.** The mixup data augmentation draws (x_1, y_1) and (x_2, y_2) at random from data distribution p^*, where y_1 and y_2 are one-hot encoded target labels, and constructs

$$\tilde{x}_\lambda = \lambda x_1 + (1 - \lambda)x_2 \tag{4a}$$
$$\tilde{y}_\lambda = \lambda y_1 + (1 - \lambda)y_2. \tag{4b}$$

The value of λ is drawn at random from Beta distribution $\mathcal{B}e(\alpha, \alpha)$ with α fixed (*e.g.*, 0.1). The training objective is the expected loss over all such mixup examples:

$$\mathbb{E}_{(x_1,y_1)\sim p^*}\mathbb{E}_{(x_2,y_2)\sim p^*}\mathbb{E}_{\lambda\sim\mathcal{B}e(\alpha,\alpha)} l(\tilde{x}_\lambda, \tilde{y}_\lambda), \tag{5}$$

where $l(x, y)$ is the loss function of neural network predictions with input x with respect to the target y. We will show that in the case of cross-entropy loss l, it can be reformulated without using label y_2, *i.e.*, not mixing labels. Therefore, even unlabeled data may be used for x_2 in the reformulation.

a) Show that the expected mixup loss (5) equals

$$2\mathbb{E}_{(x_1,y_1)\sim p^*}\mathbb{E}_{(x_2)\sim p^*}\mathbb{E}_{\lambda\sim\mathcal{B}e(\alpha,\alpha)}\lambda l(\tilde{x}_\lambda, y_1). \tag{6}$$

Hint: you will need:

- Linearity of the cross-entropy function to show that $l(x, y)$ is linear in y;
- Symmetry of Beta distribution: $\lambda \sim \mathcal{B}e(\alpha, \alpha) \Rightarrow (1 - \lambda) \sim \mathcal{B}e(\alpha, \alpha)$;
- Symmetry of the expected loss with respect to swapping (renaming) (x_1, y_1) and (x_2, y_2).

b) Prove that $2\lambda p_{\mathcal{B}e(\alpha,\alpha)}(\lambda) = p_{\mathcal{B}e(\alpha+1,\alpha)}(\lambda)$ and use it to simplify the result. *Hint:* you will need:

- Density of Beta distribution: $p_{\mathcal{B}e(\alpha,\beta)}(\lambda) = \lambda^{\alpha-1}(1 - \lambda)^{\beta-1}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}$;
- One of the defining properties of Gamma function: $\Gamma(\alpha + 1) = \alpha\Gamma(\alpha)$.