Assignment 5: Normalization and Advanced Regularization
MA-INF 2313: Deep Learning for Visual Recognition

Theoretical Task Due Date: 10.1.2021
Practical Task Due Date: 10.1.2021

Assistants: Jan Müller, Leif Van Holland

1 Theoretical Task (20 pts)

a) Mean square error and estimators *(7 points)*
In previous exercises we used the mean square error as a loss function for our model f with parameter θ, and introduced the notion of l_2-regularization. However, we did not justify the use of the MSE in a binary classification setting. In this task you will connect the MSE to both the maximum likelihood estimation and the maximum a posteriori estimation, which indicates that it is reasonable to choose the MSE as a loss.

$$\underbrace{\frac{1}{m}\sum_{i=1}^{m}\|f(x;\theta)-y^{(i)}\|^2}_{\text{mean square error}}+\underbrace{\frac{1}{2}\|\theta\|^2}_{l_2-\text{reg.}}$$

Assume that we have i.i.d. data samples $\{(x^{(i)},y^{(i)})\}_{i\in[1:m]}$ and that our data is described by a Gaussian model $p(y \mid x,\theta)=\mathcal{N}(y \mid f(x;\theta),\sigma^2)$ where σ^2 is known.

- *(2 points)* Proof that minimizing the MSE (without l_2-regularization) is equivalent to maximizing the log likelihood $\log p(y \mid x,\theta)$.
- *(3 points)* Proof that minimizing the MSE with l_2-regularization is equivalent to maximum a posteriori estimation with a Gaussian prior on the weights.
- *(2 point)* The previous statements connected the MSE to the maximum likelihood estimation as well as the maximum a posteriori estimation, which suggest that there is a connection between those two. Proof that maximum likelihood estimation is a special case of maximum a posteriori estimation.

b) Cross entropy loss and label smoothing *(8 points)*
While the mean square error can be used for binary classification problems, we use the cross entropy as a loss for the multi-class classification problem. We should convince ourselves that the minimization of this objective function also corresponds to the optimization of an estimator.

- *(2 points)* Consider our CIFAR-10 setting (softmax output, and cross entropy loss) and and develop a model which expresses our setting using probability distribution over class labels.

- *(3 points)* Proof that minimizing the cross entropy (without label smoothing) is equivalent to maximum likelihood estimation by using your model.
- *(3 points)* Your are given a noise distribution u over the class labels and you want to classify a training example according to u with a probability ϵ. Extend your probability model to include the so called label smoothing, and show that label smoothing can be considered as a regularization with the cross entropy between u and your models label distribution. Furthmore, how does the combined loss look like if u is a uniform distribution.

c) Batch normalization *(5 points)*

Consider the single layer Multi-Layer Perceptron (MLP) given in Figure 1 with two input units x_1, x_2, two output units o_1, o_2 and the shared bias b_1. We apply batch normalization in between the output units o_1 and o_2 and the Softmax activation function and then compute the total error between the predicted class probabilities and the labels p_1, p_2 with the mean square error.

1. *(2 points)* Compute the total error of the training batch in Table 1 and compute the gradient values for x_1, x_2, and b_1 **without batch normalization** between the output and the activation function.
2. *(3 points)* Compute the total error of the training batch in Table 1 and compute the gradient values for x_1, x_2, and b_1 **with batch normalization** between the output and the activation function.
3. *(1 points)* Explain who batch normalization can be applied in a setting where the training batches are given to you in an online fashion. (You have no prior knowledge about the data, you can only access the current batch and you cannot store the batches.)

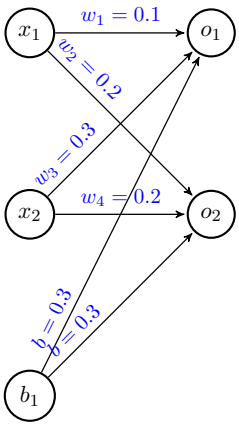

Figure 1: Multi-Layer Perceptron

n	x_1	x_2	p_1	p_2
1	0.1	0.4	0.1	0.9
2	0.8	0.2	0.95	0.05
3	0.6	0.5	0.4	0.6
4	0.3	0.9	0.75	0.25
5	0.3	0.5	0.9	0.1

Table 1: Training batch

2 Programming Task (15 pts)

The goal of this weeks programming exercise will be to further improve the accuracy of our CIFAR-10 classifier. Our baseline classifier is a MLP with two hidden layers which each have 512 hidden

units, and ReLU activation functions. The classifier is trained for 25 epochs to minimize a cross entropy loss using the Adam optimizer with a learning rate of $\eta = 0.005$ and a first and second momentum of $\rho_1 = 0.9$ or $\rho_2 = 0.95$. In each epoch the model is optimized on the shuffled CIFAR-10 dataset where we use a batchsize of 64. Your task is to incorporate the following three techniques into your model or training and tweak the parameters to push the accuracy that is achievable with our relatively simple MLP classifier.

a) Orthogonal matrix regularization (*5 points*) A orthogonal matrix W is a square matrix with the additional property of $W^T W = W W^T = I$. In this 2018 paper the authors claim that networks benefit from weight matrices which are regularized to be orthogonal. We want to make use of these benefits in out network by applying a "soft orthogonality regularization"

$$\lambda \| W^T W = I \|_F^2.$$

1. Apply the soft orthogonality regularization only to the weight matrices of your MLP (even if these matrices are not necessarily square matrices).
2. Identify a good value range for λ and plot the loss and accuracy over 25 training epochs when using the orthogonality regularization. Compare the results to the baseline network.

b) Label smoothing (*5 points*) During the last lecture you learned about different techniques to augment the training data with the goal to increase the generalization performance. One of the techniques is to apply noise to the output. In the context of our multi-class classification problem this technique is called "label smoothing" Your task is to implement a loss module with performs the following label smoothing. Let x be the linear output of your model, and $p(x) \in \mathbb{R}^d$ be the class probabilities obtained from the softmax function. A label smoothing version of the cross entropy loss given a parameter ϵ can be defined as:

$$(1-\epsilon) \underbrace{-\log(p(x))_i}_{nll-loss} + \frac{\epsilon}{d} \sum_{j=1}^{d} \log(p(x))_j \tag{1}$$

1. Implement a class called `LabelSmoothingLoss` which inherits the class `torch.nn.Module` and whose constructor takes ϵ as its input.
2. Implement a member function `forward(self, logits, targets)` which expects the linear output of your model and the target labels as its inputs and returns the smoothed cross entropy loss defined in Equation 1.
3. Train your network to minimize your "label smoothing cross entropy" loss for 25 epochs with $\epsilon \in \{5 \cdot 10^{-1}, 10^{-1}, 10^{-2}, 10^{-3}\}$. Plot the loss and accuracy over the epochs for each choice of ϵ, and compare it to the baseline model.

Hints: You can use the `log_softmax` function to compute $log(p(x))$ and the `nll_loss` function to compute the true loss value. Additionally, make sure that you use mean reduction along the batch dimension.

c) Batch normalization (*5 points*)

Batch normalization is probably one of the most popular techniques to improve a networks training's behaviour and generalization performance. Commonly these advantages are explained by arguing that batch normalization reduces covariate shift. The authors argue that batch normalization reduces the change of distribution of network activation's caused by a change of the network parameters during training (they named this change Internal Covariate Shift). However, a more recent study concludes that the reduced covariate shift might not be the main advantages of batch normalization, and argues instead that batch normalization encourages a smooth error surface (Lipschitz continuous).

1. Add batch normalization with a learn-able affine transformation to your network. The batch normalization should be applied between the layer and the activation function. Note that you also might have to update your validation function, because we don't want to change the running first and second order estimates during the evaluation of our model.

2. Plot the loss for all training steps in epoch $1 - 5$ for your model with and without batch normalization. Compare the two loss functions and discuss the claim of a smoother error surface given your results. Report the accuarcy for both models after 25 epochs.