Name

MAP545 : Deep Learning and Optimization - Final Exam
2h - No documents allowed

All questions must be answered with phrases and explanations. Simply providing the right answer won't be enough to obtain the maximum grade. As a result, imprecise or illegible answers may result in a null score at the question. Unless specified otherwise, each question is graded between 0 and 1.

Exercise 1. Deep Learning

1. Describe the operation performed by a neuron of a vintage neural network (MLP).

   A neuron is a model that computes a linear combination of the inputs (1/3), add a bias (1/3) and apply an activation function to it (1/3).

2. Describe the perceptron algorithm and give the update equation.

   The perceptron algorithm updates the weights and bias of a single neuron. Letting w_t be the vector of parameters at iteration t, the update equation of the perceptron is, for all points (x_i, y_i) satisfying $y_i\langle x_i, w_t\rangle \leq 0$ do

   $$w_{t+1} = w_t + y_i x_i.$$

3. How can we choose the input layer and the output layer of a neural network?

   They depend on the type of input/output in the data set. The size of the input layer is equal to the number of input variables (1/3). The size of the output layer is equal to the size of the output (one in regression, two in binary classification and more in multi-class classification, 1/3) with the corresponding activation function (linear, sigmoid or softmax, 1/3).

4. What is backpropagation? Describe a naive approach and explain why backpropagration improves upon it.

   Backpropagation is a procedure to compute efficiently the gradient in a neural network (1/3). A naive approach would consist in computing each element of the gradient separately, which would take roughly a number of passes through the network equals to the number of parameters in the network (1/3). Instead backpropagation allows us to compute the gradient in one single pass (1/3). Backpropagation is not an optimization algorithm (0 if the opposite is written).

5. Can we use the perceptron algorithm to train deep neural networks?

   No, perceptron algorithm is only suited for training one layer (1) of weights and a binary output (or 1). Moreover, it does not converge when data are not linearly separable, which makes it of no use in practice (0.25).

6. What is the main benefit of using a ReLU activation function? What is the main drawback of ReLU? How can you limit this flaw?

   ReLU (positive part) allows some neurons to be nonactivated (output equal to zero) which makes overfitting harder for neural network. It is also cheap to compute and does not suffer saturation in $+\infty$ (1/3). The main drawback is that some neurons may be unactivated (dead neuron phenomenon), thus making the training of the NN harder (1/3). Adding a positive bias or considering variation around ReLU is a way to lighten this phenomenon (1/3).

7. Explain the vanishing gradient problem.

   The vanishing gradient problem occurs in deep neural networks (1/3). It corresponds to the fact that partial derivatives with respect to parameters in the first layers of the networks are smaller than that with respect to parameters in the last layers (1/3). This is due to the fact that partial derivatives are multiplied by the derivative of the activation function (which is usually smaller than one) when passing through each layer (1/3).

8. In which setting(s) can we use the 0-1 loss function to train a neural network?

   We can never use the 0-1 to train a network since its derivative is 0 almost everywhere.

9. Describe the dropout procedure with details (both train and prediction steps).

   At training time, for each mini batch, for each observation, a dropout mask is sampled, taking values in $\{0,1\}^N$, where N is the number of neurons in the layer before the dropout layer. When a value of 0 is obtained, the corresponding neuron in the previous layer is shut down: its output is set to zero (0.5). When backpropagating the gradients for this observation, the partial derivatives associated to the weights connected to this neuron are zero (as the output value of the neuron is arbitrary fixed to zero, 0.25). All the gradients computed in the minibatch are averaged and used in a (S)GD step to update the weights of the network. At test time, weights are multiplied by $(1-p)$, where p is the probability of dropping a neuron in the dropout layer. (0.25)

10. Give four *different* ways of regularizing neural networks (without explaining them).

    Dropout, batch normalization, early stopping, penalization, weight clipping are different ways of performing regularization in neural networks (see the course for details, 0.25 per answer).

11. We prefer to use CNN for image classification. Could we use vintage neural networks (MLP) instead? Justify.

    We could use classical neural networks for prediction tasks based on images. However, they would not leverage the spatial structure of the images (1/2) and they would not be able to detect the same elements at different locations in the image (1/2), plus they would have a lot more parameters.

12. In a CNN, rank the following layer types from the one that involves the largest number of trainable parameters to the one that involves the smallest number of trainable parameters: Convolutional layer, Pooling layer, Dense layer. Justify.

    The correct order is Dense layer, Convolutional layer, Pooling layer (0.25, 0 if incorrect). Dense layer has all possible connections and weights are different for each (0.25). COnvolutional layer is sparse and some weights are shared (0.25). Pooling layer contains no trainable parameters (0.25).

13. Give two differences between a convolutional layer and a pooling layer.

    A convolutional layer contains trainable parameters whereas the pooling layer does not (1/2). A convolutional layer operates on the whole depth of the input volume whereas the pooling layer operates on each feature map separately (1/2).

14. What is the operation performed by a flatten layer? How many parameters does it contain? Where is it used in CNN?

    A flatten layer simply changes the shape of the input without modifying the inner values: it allows changing the input, which is usually a set of tensors, into a (one-dimensional) vector (0.5). Therefore, it contains no parameters (0.25). It is used at the end of the network, before dense layers that operate on vectors and not on tensors (0.25).

15. Connect each neural network to one of its feature (± 0.2 per correct/incorrect association):

- GoogLeNet
- Xception
- ZFnet
- ResNet
- AlexNet

- Split computations across two GPUs
- Involves shortcut connections
- Decompose a convolution into one convolution that operates on the depth only and one on the spatial dimensions only
- Use 1x1 convolutions to reduce the number of computations
- Use deconvnet to understand feature maps

Name

16. What is the difference between segmentation and object detection?

    Segmentation consists in assigning a label to each pixel of an image (0.5), whereas object detection consists in creating a box that contain an object (0.5). Segmentation is thus more difficult to achieve than object detection.

17. What tasks can be solved with YOLO? What is the main improvement of YOLO compared to previous algorithms?

    YOLO can be used to detect object in images by automatically creating bounding boxes with associated labels and corresponding probabilities (0.5). YOLO works in an end-to-end fashion by creating bounding boxes proposals and class probabilities in the same pipeline, contrary to past approaches that first produced bounding boxes and, in a second step, predicted the object inside the bounding boxes (0.5).

18. Is it possible to consider more than one hidden layer in a RNN? Give the equation used to compute the hidden state h_t in regular (neither LSTM nor GRU) RNN.

    It is totally possible to consider more than one lyaer in RNN. In the lab sessions, we have considered RNN with two hidden layers. Note that the number of hidden layers is different from the size/dimension of the hidden state (1/2 for an example or a difference between HL and hidden state size). Plus (1/2) for the exact equation:

    $$h_{t+1} = \tanh(W_{HH}h_t + W_{HI}x_{t+1} + b).$$

19. A problem called vanishing gradient phenomenon occurs in RNN. What is it exactly?

    The vanishing gradient problem is different for RNN and for ANN. When considering a RNN, the gradient does not vanish: all components of the gradient are not close to zero (0.5). However, each component of the gradient can be written as a sum, whose each element corresponds to a particular time step. It turns out that the terms corresponding to the inputs at the beginning of the sequence are, by computation, very close to zero (0.25). It means that the network 'forgets' that its parameters play a role in the first hidden states of the sequence, therefore leading to forgetting the long-term dependencies in the sequence (0.25).

20. What is truncated backpropagation? What is the interest of using such a procedure? Give a potential drawback.

    Truncated backpropagation is used to compute the gradient of the loss in RNN. It works by considering a small subsequence of the original sequence, computing the loss, backpropagate the gradient and updating the weights. Then, the second part of the sequence is fed to the network and the procedure iterates (0.5). It helps to decrease the number of computations needed for one single iteration (0.25). Unfortunately, it increases the long-term memory problem, by splitting the whole sequence into smaller subsequences (0.25).

21. Describe the Stochastic Gradient Descent algorithm.

    Start from an initial point, for example $w_0 = 0$. At each iteration, sample a minibatch $I \subset \{1, \dots, n\}$ and do

    $$w_{t+1} = w_t - \eta \nabla f(w_t),$$

    where η is the learning rate (set in advance) and f the function to minimize.

22. What is the mathematical intuition behind gradient descent update? What is the main mathematical assumption required to make Gradient Descent consistent?

    By computing the first order approximation of the function at the current iteration, we see that the direction $-\nabla f$ is the direction that minimizes this approximation. So locally, the gradient descent direction is the one that minimizes the function (0.5). In order to converge, the function to minimize must be convex (0.25) and the gradient must be computable ($f \in C^1$, 0.25).

23. Assume that we can prove that for any function f in a class of functions, after k steps of an algorithm, we have $f(x_k) - f(x^*) \leq \rho^k \|x_0 - x_*\|^2$, for x_k the output and x^* the minimum of the function, x_0 the initial point. If I have reduced my error by factor 10^{-3}, compared to the starting point, after 200 iterations, how many more iterations would be necessary to complete a reduction of 10^{-9}?

    After 200 iterations, the error is reduced by a factor 10^{-3}. Thus, we have $\rho^{200} = 10^{-3}$ (1/3). In order to reduce the error of a factor 10^{-9}, we need to have a number of iterations satisfying $\rho^k = 10^{-9}$ (1/3), which is verified for $k = 600$. Therefore, it takes 400 more iterations to reach a reduction of 10^{-9} (1/3).

24. Give an algorithm (other than Gradient Descent) that provides such a rate for a class of function (to be precised). What is the factor ρ?

    For μ-strongly convex functions (0.33), SAG algorithm (0.33) verifies this rate with $\rho = 1 - \min\left(\mu, \frac{1}{n}\right)$ (0.33). Same answer for SVRG and SAGA.

25. What is the intuition for the choice of the learning rate in SGD? What step-size sequence $(\gamma_t)_{t\geq 0}$ is typically used?

    SGD introduces randomness into the optimization process via the sampling of minibatches. Reducing the noise in SGD requires to decrease the step size so that ultimately the algorithm converges to a deterministic point (0.5). For convex and L-smooth functions (0.25), one can choose $\gamma_t = 1/\sqrt{t}$ (0.25). For strongly convex functions, one can choose $\gamma_t = 1/t$.

26. What does good/bad conditioning correspond to? Give a mathematical definition of the condition number. What are the 2 main techniques well suited to improve convergence for a poorly conditioned convex function?

    A bad conditioning means that the largest eigenvalue of the Hessian matrix is much larger than the smallest eigenvalue (0.33). The conditioning number is given as a ratio of these values. It stands for the difficulty of the optimization problem (0.33). Newton's method or (S)GD with momentum are suited for poorly conditioned problems (0.33).

27. We consider the case where $f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$ is convex and smooth but **not** strongly convex. We want to minimize f over $\mathbb{R}^d$. Complete the following comparison of SGD and GD

	GD	SGD
Convergence rate after k iterations	$\frac{1}{k}$	$\frac{1}{\sqrt{k}}$
Complexity per iteration	$O(nd)$	$O(d)$
Number of iterations to reach precision $\frac{1}{n^{1/4}}$	$n^{1/4}$	$n^{1/2}$
Number of iterations to reach precision $\frac{1}{n^3}$	n^3	n^6
Total complexity to reach precision $\frac{1}{n^{1/4}}$	$n^{5/4}d$	$n^{1/2}d$
Total complexity to reach precision $\frac{1}{n^3}$	$n^4 d$	$n^6 d$

28. How would you summarize the conclusion of the last two lines of the previous table?

    SGD has a lowest complexity for reaching a small precision (smaller than $1/n$) compared to GD. However, when a high precision is needed (larger than $1/n$), GD is more efficient than SGD (0.75). In Machine Learning, we often want to minimize the empirical risk, which is a noisy version of the true risk. Therefore, we often do not care about a very high precision (the quadratic risk is typically of order $1/n$) (0.25).

29. What is the main idea behind Nesterov Accelerated Gradient Descent? Is it a first or second-order method?

    The main idea behind Nesterov acceleration method is to remember the descent direction and use it in the current iteration. It enforces the new direction to be close to the previous one, thus creating a notion of velocity in the GD algorithm (0.5). It is a first-order method since it only uses the gradient of the function (0.5).

30. What is the intuition behind variance-reduced algorithms? Give an example of such an algorithm, and describe it.

The intuition behind such algorithms is to introduce a bias in the descent direction and to decrease the variance. The variance results from the sampling of minibatches. Practically, it is based on the use of past gradient information in the new iterations (0.33). SVRG, SAG and SAGA are such algorithms (0.33). See course for the algorithm (0.33)

31. Give one advantage and one drawback of SVRG, compared to SAG.

SAG works by storing the gradients at previous iteration points, which is of order $O(nd)$. It presents memory issues compared to SVRG, which only requires storing one gradient and two vectors, that is $O(d)$ (0.5).

However, the cost complexity of SVRG is twice that of SAG since it requires computing two different gradients at two different points (0.5).

32. Give the update equation in Newton's method. What happens when this method is applied to quadratic functions?

The update equation of Newton's method is

$$w_{t+1} = w_t - (\nabla^2 f(w_t))^{-1} \nabla f(w_t).$$

If this method is applied to quadratic function, it converges in one iteration. Indeed, if

$$f(w) = \frac{1}{2} w^T A w + b^T w + c,$$

for A invertible and symmetric, with positive eigenvalues, we have

$$\nabla f(w) = Aw + b \quad \text{and} \quad \nabla^2 f(w) = A.$$

Note that the minimum of f is reached at $w^\star$ satisfying $\nabla f(w^\star) = 0$, that is $w^\star = -A^{-1}b$. Thus, starting from $w_0 = 0$, we have

$$w_1 = w_0 - A^{-1}b = A^{-1}b = w^\star,$$

which proves the convergence of Newton's method in one iteration for quadratic functions.

33. **(3 points)** We consider an optimization problem with the following characteristics: we want to solve

$$\arg\min_{w \in \mathbb{R}^d} \left\{ F(w) := \frac{1}{n} \sum_{i=1}^{n} f_i(w) \right\}$$

over $\mathbb{R}^d$. 1) What does f_i correspond to in ML? 2) Can you give a precise example of a problem which results in d vary large, n very large, f_i being non-convex, non smooth.

1) The functions f_i correspond to the loss associated to the ith observation (0.5).

2) The quantity d corresponds to the number of parameters in the considered ML technique. For deep neural networks, trained on a large data set, we have n, d very large. Moreover, since there are more than one hidden layer, the problem is nonconvex and nonsmooth if the weights are not clipped.

For each of the following methods, explain why it is well suited or not and propose a ranking.

Method	Advantage	Drawback	Ranking
LBFGS	Efficient for poor conditioned problems	$O(d^2)$	3
RMSprop	adaptive step size per coordinates	...	1
SAGA	Efficient for large n	Variance reduction not efficient for large d	2