MACHINE LEARNING 1

1) Which of the following options are true?

It is always better to have a learning rate that is too high than too low
Adding a moment rate accelerates learning <-
The minimum reached during the learning process is always the same
A multilayer neural network allows to approximate any function (it is a universal approximator) <-

2) Why do we prefer ReLU instead of the tanh or the sigmoid as activation function for problems with a lot of layers?

Because the tanh and sigmoid derivatives depend on the function itself and the derivative of the ReLU function does not
Because the ReLU function is not differentiable at 0 and this makes the learning more stable
Because its derivative is always 1 when x > 0 and we avoid thus the vanishing gradient problem <-

3) Which of the following options are true?

A good regularization can help solving a high variance problem <-
Getting more data can help solving a high variance problem <-
Training a bigger network can help solving a high bias problem <-
Training a bigger network can help solving a high variance problem

4) The main difference between neural networks and extreme learning machines is that in the latter the parameters can be learned analytically (no iterative learning)

False
True <-

5) The surface error contains more local minima when..

there is a high number of hidden layers <-
it is independent on the amount of hidden layers
there is a low number of hidden layers

6) If we apply 15 filters (3x3) to an image of dimensions 20x20x30 (n_w x n_h x n_c). How many parameters must learn this convolutional layer?

4065 <-
4100
4050

7) The backpropagation algorithm estimates..

the error between the observed and predicted values
the learning rate
the weights of the network
the derivatives of the cost function with respect to the weights of the network <-

8) To avoid overfitting and regularize the network we can..

monitor the training and validation losses and retrain the network to an epoch were both error curves are similar <-
include L1 or L2 penalty terms in the cost function <-
add more layers to the network
apply early-stopping <-
set all the activation functions to sigmoidal to limit the output values to the 0-1 domain

9) Can an application be trained to remove noise with a variational autoencoder?

Only if we apply additional restrictions in the latent space
Yes <-
No

10) Early-stopping consists on…

to stop the learning process according to a stopping criteria applied over the error on the validation dataset (independent of the training dataset) <-
it is not a method to avoid overfitting
stopping the learning process when the error in the training dataset reaches a minimum, with the objective to avoid overfitting by reducing the net size (neurons and layers)

11) Indicate which of the following sentences regarding the neural networks are true:

Overfitting is not a matter of concern in these models
They are very interpretable (the relations learned in the model among variables are easy to understand)
They can solve a quite variety of problems: classification and regression tasks <-
The parameters are learned in an iterative process, and not learned analytically <-

12) When does it make sense to do transfer learning from A to B?

When you have much more data in A than in B <-
When A and B have very different inputs
It is always a good idea

13) Which of these are specific characteristics of the convolutional neural networks?

When we use them on images they act as an image feature extractor with a certain hierarchy <-
They need to have a huge amount of filters in each of the layers
Only the last layer is trained
The pattern learnt are translation invariant <-

14) Why are residual blocks useful?

They help avoiding the exploding gradient problem since they easily learn the identity function
They help avoiding the training error degradation in a very deep neural network since they easily learn the identity function <-
They help avoiding the high variance problem since they easily learn the identity function

15) If we have a small dataset, which of these methods is better for the gradient descent?

Mini-batch gradient descent
Stochastic gradient descent
Batch gradient descent <-

16) Given a feature map of dimensions n_w x n_h x n_c, where n_w is the width, n_h the high and n_c the number of channels. Why do we apply a pooling layer?

To reduce n_w x n_h <-
To reduce n_c
To reduce overfitting layer by layer

17) We pretend to solve a classification task with 100 input variables (and only 1 output variable, the class) with a multilayer feedforward neural network of two hidden layers, with 20 and 10 hidden neurons, respectively. ¿How many parameters does the net has to learn?

more than 2200 <-
less than 2000
between 2000 y 2200

18) If we increase the value of the learning rate..

we descend slower to the local minimum and we tend to oscillate closer/stable around the minimum when we reach it
we descend slower to the local minimum and we tend to oscillate farther/unstable around the minimum when we reach it
we descend faster to the local minimum and we tend to oscillate farther/unstable around the minimum when we reach it <-
we descent faster to the local minimum and we tend to oscillate closer/stable around the minimum when we reach it

19) What happens if all the activation functions (including those in the hidden layers) are linear?

You obtain nonlinear relationships between the explanatory and the response variables though not very nonlinear
It fits better the data
This is equivalent to a linear model <-

20) For classification tasks it is best to..

minimize the mean squared error with sigmoidal activation function in the output layer
minimize the negative log-likelihood of a Bernoulli distribution with linear activation function in the output layer
minimize the negative log-likelihood of a Bernoulli distribution with sigmoidal activation function in the output layer <-
minimize the mean squared error with linear activation function in the output layer

SUMMARIZE

Machine/deep learning problems:
Supervised problems (input-output data): Multilayer Networks or Feedforward Nets; Autoassociative memories (Hopfield).
Unsupervised problems (input data only): Competitive Networks; Recurrent Networks or Elman/Jordan nets (with memory).

Single-layer networks cannot approximate nonlinear problems, while multi-layers networks can.

Backpropagation: is an algorithm used to efficiently compute the gradients of the loss function with respect to each parameter (weight and bias) in a neural network. Here's a step-by-step explanation:
1. Forward pass: the input data is passed through the network, layer by layer, to produce an output. This output is then compared to the target using a loss function, which quantifies the error.
2. Backward pass (Backpropagation): starting from the output layer, the algorithm computes the gradient of the loss with respect to the network's outputs. Using the chain rule, these gradients are propagated backward through each layer, calculating the gradients with respect to the weights and biases. This process continues layer by layer until the gradients for all parameters are obtained.
3. Weight update: the calculated gradients are then used to update the parameters (typically using an optimization algorithm like gradient descent), which minimizes the loss.

Deep Learning: refers to neural networks that have multiple layers. The term "deep" indicates that the model has several consecutive layers, each one transforming the input into more abstract features. Essentially, the "depth" of a model is determined by the number of these layers, allowing the network to learn complex representations of the data.
Learns hierarchical representations and features directly from the data, reducing the need for manual intervention.
Generally requires large volumes of data to capture complex patterns effectively.
Typically demands significant computational power (GPUs/TPUs) for training, due to the complexity and number of parameters.
Models tend to be more of a "black box," making interpretation and explanation of decisions more challenging.
All parameters (weights, biases, etc.) are typically updated simultaneously during each training iteration. Using techniques like gradient descent with backpropagation, once the gradients of the loss function with respect to each parameter are computed (often

for a whole batch or mini-batch of data), every parameter is adjusted concurrently. This simultaneous update allows the model to learn in an integrated manner, capturing the complex interdependencies among the parameters.

Gradient descent: is an optimization method where weights are updated in the opposite direction of the error gradient. In other words, the weights are adjusted based on the gradient of the error function.
$z = w^T x + b \quad \rightarrow \quad a = \sigma(z) \quad \rightarrow \quad L(a, y)$ where L is the loss function and σ the activation function (sigmoid in this case).
To minimize this function we need to find the minimum with respect to the weights and biases dL/dw, dL/db. Dimensions: x(n, 1), w(h, n), b(h, 1) where h is the number of neurons.
Minimizing the loss function corresponds to maximizing the probability of the correct class.

Logistic regression: used for binary classification. As activation function utilizes the sigmoid function to output a probability between 0 and 1. As output predicts the likelihood of one class versus the other (e.g., 0 or 1). The loss function is often optimized using binary cross-entropy loss.

Softmax regression: generalizes logistic regression for multi-class classification. As activation function applies the softmax function, converting raw scores (logits) into probabilities that sum to 1. As output provides a probability distribution over multiple classes. As loss function typically uses categorical cross-entropy loss for training.

Activation function: sigmoid is prone to the vanishing gradient problem in deep networks because its derivative becomes very small for large positive or negative inputs, slowing down learning.
Tanh faster learning than sigmoid, has a steeper derivative than sigmoid, allowing for a bit more effective gradient flow. Still saturates at both ends, which can result in vanishing gradients when used in many layers.
ReLU avoid the problem of the vanishing gradient because the derivate is 1 for x>0. Faster in learning, less computationally expensive. Promotes sparsity by outputting 0 for all negative inputs, which can lead to a more efficient and less overfitted network. As disadvantages, it's not differentiable in 0, although in practice this rarely causes issues. Can lead to the "dying ReLU" problem: if a neuron outputs 0 for most inputs, it may stop updating its weights, causing parts of the network to become inactive. Problems that lead to the dying ReLU problem: if the learning rate is too high, the weight updates can be excessively large, causing weights to swing dramatically—even becoming very negative. This instability can lead to divergence during training. If the bias becomes very negative it can push the inputs of the ReLU into the negative range.
Leaky ReLU 0.01x when x<0.

ReLU (or Leaky ReLU) for all hidden layers.
Tanh or sigmoid only in the final layer for binary classification problems.
Softmax in the final layer for multi-class classification problems.

Datasets: Training: train multiple models using the training set. Development (Validation): evaluate these models on a development set and select the one that performs best. Testing: test the chosen model on the test set to obtain an unbiased estimate of its performance on unseen data.

Hight bias: underfitting, change network structure or bigger, more training time needed.
High variance: overfitting, more data needed, regularization, change network structure.

Regularization L1: adds a penalty term to the loss function that is proportional to the sum of the absolute values of the model's weights.

Regularization L2: adds a penalty term to the original loss function, which is proportional to the sum of the squares of the model's weights. This penalty discourages large weight values, encouraging the model to keep them small. This helps to prevent overfitting and improves generalization on unseen data. The hyperparameter λ must be tuned (dev. dataset) carefully. A very high λ can lead to underfitting, while a very low λ may not sufficiently mitigate overfitting.

Regularization Dropout: randomly "drops" (i.e., sets to zero) a fraction of the neurons and their connections during training. During each training iteration, a predefined percentage of neurons is temporarily deactivated. This forces the network to not rely too heavily on any one neuron, encouraging it to develop redundant representations. By randomly disabling neurons, dropout helps prevent overfitting, as the network learns more robust features that generalize better to unseen data.

Batch gradient descent: calculates the gradient of the loss function using the entire dataset before updating the parameters. This provides a stable and accurate direction for the update but can be very slow and computationally expensive with large datasets.

Stochastic gradient descent (SGD): updates parameters for each training example individually. This leads to noisy but frequent updates, often resulting in faster convergence on large datasets, although it may cause fluctuations in the loss.

Mini-batch gradient descent: divides the dataset into small batches and updates parameters after processing each batch. It strikes a balance between the stability of batch gradient descent and the speed of SGD, making it widely used in practice.

Gradient descent with momentum: it builds up "momentum" that helps smooth the descent path, reducing oscillations and accelerating convergence, especially in ravines or when facing noisy gradients.

Convolutional Networks (ConvNets): are able to learn local patterns, contrary to the dense ones that learn global patterns. In case of images, these patterns are contained in 2D filters. These patterns are translational invariant, it means that the pattern can be recognized wherever it is the image (right, left, bottom, up). That's why ConvNets are particularly useful in image recognition. Moreover they need less images to be trained with respect to dense networks. One more advantage is the spatial hierarchy learning.

Filters: filters size is usually odd to maintain one pixel in the center

Padding: is the process of adding extra pixels (usually zeros) around the borders of an input image. This is done to control the spatial dimensions of the output feature maps after applying convolutional filters. Padding can help maintain the same width and height of the image after convolution. By adding a border, padding allows the convolutional filter to fully cover the edges of the image, ensuring that features near the borders are not neglected.

$$n \times n \ * \ f \times f \quad \rightarrow \quad (n + 2p - f + 1) \ * \ (n + 2p - f + 1)$$

Stride: refers to the step size by which the convolution filter moves across the input image.

$$n \times n \ * \ f \times f \quad \rightarrow \quad \left[\frac{(n + 2p - f)}{s} + 1\right] \ * \ \left[\frac{(n + 2p - f)}{s} + 1\right]$$

If it is not an integer, it is rounded down to the nearest integer.

Convolutional layers: the frontal plane of the parallelepiped depends on the convolution variables (p, s, f, etc.). The depth is equal to the filter size in each layer

Pooling: is a downsampling operation used in convolutional neural networks to reduce the spatial dimensions (width and height) of feature maps. Can be max, selects the maximum value in each region; or average, computes the average value in each region. No parameter to learn.

Residual Networks (ResNet): used mainly when extreme deep neural network to avoid the problem of the vanishing gradient. This architecture is based on residual blocks that are basically skipped blocks of layers (skip connection). Another pro of this network is related to the wrong learning, in fact some intermediate layers can learn in a wrong way, so that skipping some layers can affect positively the learning.

Transfer Learning: leverages a model pre-trained on a large dataset and fine-tunes it on a related task with limited data. Transfer of knowledge from A (more info/data) to B (less info/data). Example: style transfer, that uses a pre-trained CNN to separate the content of one image from the style of another, then optimizes a new image to blend both aspects.

Generative Adversarial Networks (GAN): consist of two neural networks—the generator and the discriminator—trained in an adversarial setting. The generator creates synthetic data aiming to resemble real data, while the discriminator evaluates whether a sample is real or generated. The networks compete, with the generator improving until its outputs become indistinguishable from real samples.

Autoencoder (AE): self-supervised algorithm (labels are automatically generated from inputs). The objectives are to learn a compression representation of the data and reconstruct the input after having compressed it. This algorithm is made of: an encoder function (to compress, convolutional layers, pooling layers), a decoder function (to decompress, transposed convolutional layers, unpooling layers) and a loss function.
Good at learning representations.

Variational Autoencoder (VAE): instead of learning an arbitrary function to encode the input, you are learning the parameters of a probability distribution (ex. Gaussian) modeling your data. Constraints are added. Latent space more structured.
Inference: sampling of points from this distribution, passing them through the decoder function and getting new data samples (generative models).
Training: prediction of variance and mean, sampling from the normal distribution associated and generating output.
Loss term: can be a reconstruction term as in AE (ex. MSE or cross-entropy); or KL divergence which measures how different the encoder's learned distribution, $q(z|x)$, is from the standard normal distribution $p(z)$. Keeping this divergence small encourages the latent space to be "organized" like a standard Gaussian, which makes it easier to generate new samples. However, if the KL Divergence is pushed too low, the model may overly compress the latent space and harm reconstruction quality. Conversely, focusing only on minimizing the reconstruction loss could lead to a disorganized latent space, making it harder to produce realistic samples. Striking the right balance between these two losses ensures both good reconstructions and a well-structured latent space for generation.
Good at generating new samples.

Recurrent Neural Networks (RNNs): is a type of neural network specifically designed to handle sequential data (such as text, time series, or speech) by maintaining a form of "memory" through its recurrent connections, because they process inputs one at a time while retaining information from previous steps. Unlike feedforward networks, RNNs have loops that allow information from previous time steps (hidden states) to influence the current output. This helps the network capture temporal dependencies.
Standard RNNs can suffer from issues like vanishing or exploding gradients, which make it hard to learn long-term dependencies.

Long Short Term Memory (LSTM): has been developed to mitigate the problems related to the RNNs. LSTM units contain memory cells that store information over long periods. They use gates (input, forget, and output) to regulate the flow of information.

Comparison between CNNs and RNNs: CNNs are designed for spatial data (like images). They use convolutional filters to extract local features and pooling layers to reduce spatial dimensions.

RNNs are designed for sequential data (like text or time series). They process input one time step at a time, maintaining a hidden state that captures information from previous steps.
CNNs have feedforward connections where each layer operates on the feature maps produced by the previous layer. Weight sharing in the filters helps in recognizing patterns regardless of their position.
RNNs their feature recurrent connections that allow the network to "remember" past information. Each step's output is influenced by previous inputs due to these loops.
CNNs, convolutional operations can be executed in parallel across different spatial locations, making them computationally efficient.
RNNs, computations are sequential because the processing of a current time step depends on the previous one, which limits parallelization.

Diffusion models: are generative models that gradually add noise to data until it becomes random, and then learn to reverse this process to generate new, high-quality samples from noise.

Reinforcement learning: is a machine learning paradigm where an agent learns to make decisions by interacting with an environment. The agent takes actions, receives rewards or penalties, and adjusts its behavior to maximize cumulative rewards over time.