Exam of
Introduction to Machine Learning (TC0)
18 décembre 2020 – 3h

DON'T RETURN THIS SHEET BEFORE YOU ARE ALLOWED TO DO SO!
(you can write your name on the copies in the meantime)

General advice:

- Authorized documents: unlimited personal notes.
- Do not hesitate to do the exercises in any order you like: start with the ones you feel are quick to deal with.
- When you are allowed to start, before you start the first exercise, go through the subject quickly. In each exercise, the most difficult question is not necessarily the last, please feel free to skip some questions. Don't hesitate to go and scrape off points where they are easy to take. (vous pouvez "aller grapiller les points")
- The grading points (scale) is indicative, if the exam is too long a correction factor will be applied. So don't panic in front of the length, what you do, do it right! Also, you may notice that points sum up to 25 (6+8+6+5) instead of 20, so, I will do *something*.
- French: Vous êtes autorisés à composer en Français. (Y compris en insérant des mots techniques comme overfitting ou regularization en anglais quand vous ne savez pas la traduction).
- French: si certains bouts de l'énoncé ne sont pas clairs, je peux les traduire ! N'hésitez pas à demander si vous n'êtes pas sûrs.
- Calculators not allowed (and useless). No electronic device allowed (cell phone, etc).
- At the end, we will collect your papers. You must return this subject to us, together with your paper. You can leave after you have returned your copy.
- It is forbidden to lick the paper to stick it. Anonymization is canceled this year. (covid..)

DON'T RETURN THIS SHEET BEFORE YOU ARE ALLOWED TO DO SO!
(you can write your name on the copies in the meantime)

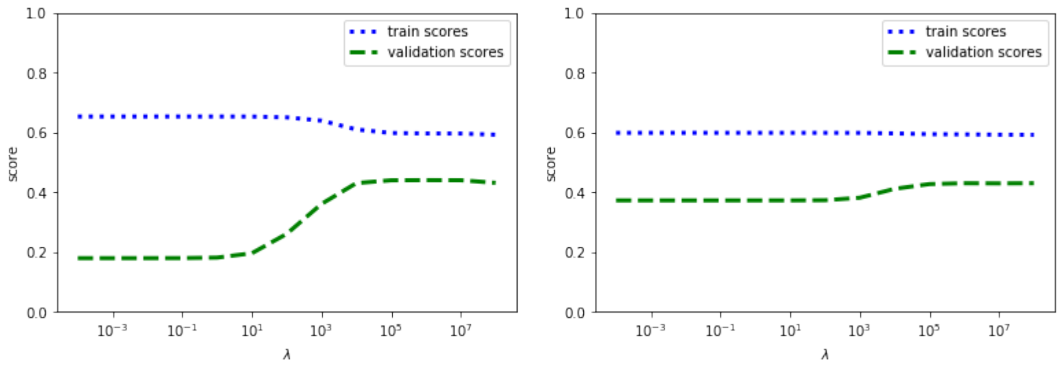

Figure 1: Left: Case a; Right: case b

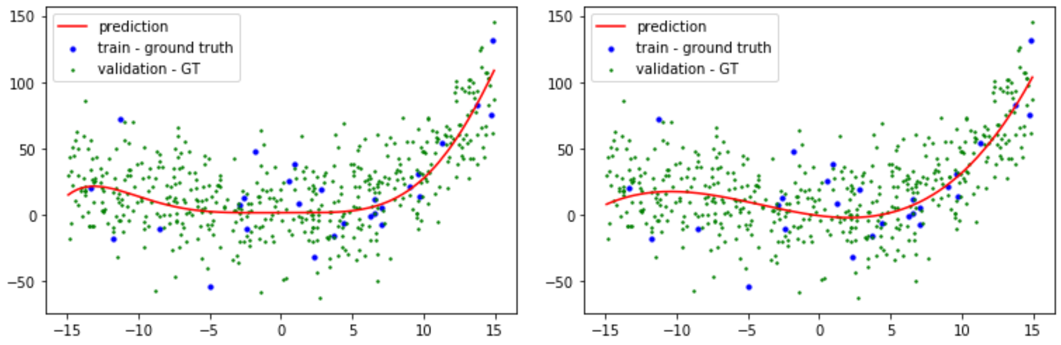

Figure 2: Left: Case c; Right: case d

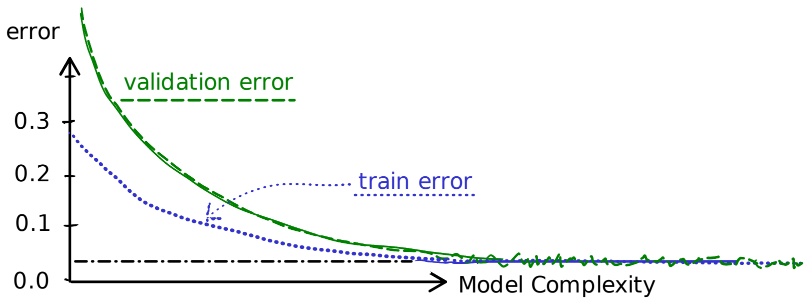

Figure 3: Model complexity increases from left to right.

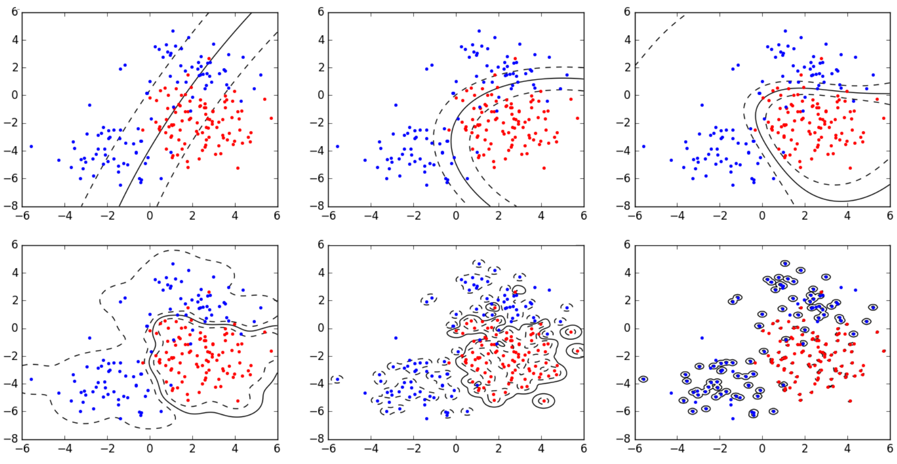

Figure 4: A hyperparameter is increased, from left to right, top to bottom.

1 Analysis of experimental results (6 points)

1. See figure 1. We perform some regression task with regularization controlled by λ. We have two models, one being more complex (more expressive) than the other.
   (a) Which one is the richer, more expressive model, the left one or the right one ? How do you explain the variations of the curves in case (a) (left)? (1 pt)
   (b) Which model do you think is best to fit your data? Why? (Remember that the score may not fully characterize how well the data is fitted). (0.5 pt)
2. See figure 2. We are in a similar setup as question 1, although it's not the same model/hyper-parameters, etc. In cases (c) and (d), one of the two is a complex model with high regularization, and the other is a less complex model with no regularization. Can you tell which is which? Explain your reasoning. (0.5 pt)
3. See figure 3.
   (a) What has happened? Is there overfitting? (0.25 pt)
   (b) Why isn't the train error 0? Why isn't the validation error 0? (0.5 pt)
   (c) What amount of test error can you reasonably expect? (0.25 pt)
4. See figure 4. A hyperparameter is increased, from left to right, top to bottom. The model is an SVM with a Kernel.
   (a) How do you interpret the solid line? (0.5 pt)
   (b) How do you interpret the two dashed lines? (0.5 pt)
   (c) Which cases seem to be in a situation of overfitting? Which cases seem ok? (0.5 pt)
   (d) Knowing nothing about the data, which choice of hyper-parameter would you pick (assuming you cannot see the validation score, or assuming it's very similar in all the "good cases")? Why? (0.25 pt)
5. Benji has a lot of ideas. He writes ML papers, using a classic and interesting (difficult enough) dataset, called ImageNet. In his first paper, he does the train/test split (80-20), he does cross-validation on the train part (the 80%), and he uses the test set only after all hyper-parameters have been optimized.
   Several months later, he has a new idea, builds an improved network, optimizes the hyper-parameters, and tests on the 20% of the test set. He gets a better performance.
   Several months later, he has yet a new idea, builds again an improved network, optimizes the hyper-parameters, and tests on the 20% of the test set. He gets yet a better performance.
   What could be wrong with Benji's work? (0.5 pt)
6. We have an algorithm that automatically recruits people, based on their résumés (Curriculum Vitae). It's form Amazon and we discover it's quite racist and sexist, because the database it's trained from also has these biases. We assume that we can quantitatively asses how much the training data is racist/sexist. What could we try to do to correct the identified problems? (Remark: in real life, it's more complicated, but ok, this is an illustration). Explain in 2-3 lines maximum. (0.75 pt)

2 Linear Regression - exercise very closely related to the Lecture's material (8 points)

1. Describe the linear regression algorithm, when it's performed without regularization, and the solution is found using Gradient Descent (not the exact analytical solution that can sometimes be achieved).
   You will start by describing the kind of task that linear regression is made for, then define the cost function to be minimized (use least squares), the optimization algorithm that is used, and then write a very short pseudo-code of what the `fit` function may look like. You will also provide the corresponding `predict` function. (2 pts)

2. Let's compute an update of Linear Regression. Let's suppose the data is one dimensional: $X = (1, 1, 2, 2)$, $y = (1, 2, 2, 3)$. The initial weight vector is $w = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. Advice: write X in a matrix form. Use the learning rate $\eta = 1$. Hint: if this seems long to you, keep this computation for later (it's ok to refer to a point later in your paper). (1.5 pt)

3. If the relationship between input $\vec{x}$ and output y is not linear but has some dependence on products of the input features $x_1, x_2, \ldots$, what can we do? (0.5 pt)

4. First variation on Linear Regression. (forget the data of question 2, we go back to the general case).

   (a) For some reason, someone is interested in finding the solution to the optimization problem $J(\vec{w}, \{\vec{x}_n\}_n) = \frac{1}{4}\sum_n^N (\vec{w}\cdot\vec{x}_n - y_n)^4 + \frac{1}{4}\lambda||\vec{w}||^4$. Derive the Gradient Descent algorithm's steps. This new algorithm may be called LinReg^2 (Lin. Reg., squared). (1 pt)

   (b) Describe the change(s) needed in the `fit` and `predict` functions (comparing LinReg and LinReg^2). (0.25 pt)

   (c) We may define a sort of "MQE", mean quartic error: $MQE = \frac{1}{N}\sum_{n=1}^N([\text{error on point n}])^4$. After all, why not ? In your opinion, which algorithm will perform best in terms of the MQE, the usual LinReg with some regularization (similar to qestion 1), or LinReg^2 (similar to question 4a)? Explain why. Ideally, you may use maths notations. (0.75 pt)

5. Second variation: mini-batches. Let's forget about question 4. We recall the idea of the mini-batches. The idea is that instead of optimizing the error for all examples (training data samples) at once, we can use only a subset of them (chosen at random, or cycling through the data, for instance first using examples 1-100, then 101-201, etc).

   (a) Describe what this means, for a `fit` function, in terms of maths and pseudo-code. You may assume that the shape of $\vec{\nabla}_\Theta J$ is known and so you may use the term $\vec{\nabla}_\Theta J$ generically. (1 pt)

   (b) In practice, for complex models (e.g. for Deep Networks), it is often observed that mini-batch GD leads to better generalization than full batch GD. Can you explain intuitively why ? Hint: imagine that your training data is of good quality, but that the input space is so big that there is still "randomness" in the data, in that your training set is not very similar to the test set, simply because of the "sampling noise" (the kind of noise that makes it so that when you flip a coin 10 times, typically it does not give 5 heads and 5 tails). (0.75 pt)

   (c) Cite another potential advantage of using mini-batches instead of the "full batch" GD. (0.25 pt)

3 O-V-O classification (6 points+bonus)

In this problem, some parts are inter-dependent. However, to some extent, if you cannot do one of the parts, you can still do some of the later parts. If you don't find this easy, I would advise to keep this problem for the end.

We want to perform multi-class classification (K classes) on $(D-1)$-dimensional data. The extended data points (with a 1 as the value of the first component) will then be D-dimensional, which is very convenient. We have N examples in the training set. The data is the set of the training examples $X = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N, k_{true}^{(1)}, k_{true}^{(2)}, \dots, k_{true}^{(N)}\}$, where $k_{true}^{(n)}$ is the label of example n (i.e. it takes values in $\{1, 2, \dots, K\}$).

There are K classes. We encode the ground truth as one-hot vectors $\vec{t_n}$ of size K, or concretely $t_n = (0, ..., 0, 1, 0, ..., 0)^T$. This means that for the components t_{nk} we have $t_{nk} = \delta_{k, k_{true}^{(n)}}$, that is to say, we have $t_{nk} = \begin{cases} 1 \text{ if } k = k_{true}^{(n)} \\ 0 \text{ if } k \neq k_{true}^{(n)} \end{cases}$.

We plan to use a **one-vs-one classification scheme** (o-v-o). That means that for each pair of classes (k, k'), we will have a (binary) classifier $\vec{w}_{kk'}$. It will tell us if we are more likely to be in class k or class k'. For now, don't worry too much about the o-v-o scheme.

1. We start with the simpler $K = 2$ case, but keeping the one-hot encoding, so for class 1, $t_n = (1, 0)^T$, and for class 2, $t_n = (0, 1)^T$. The hyperplane characterized by the (extended) vector $\vec{w}_{12}$ points towards class 1: when $\vec{x}$ is on the "class 1" side of the hyperplane, then we have $\vec{w}_{12} \cdot \vec{x} > 0$. The loss function we consider has a term of the form

$$\sigma(\vec{w}_{12} \cdot \vec{x}_n) - (t_{n1} - t_{n2}) \tag{1}$$

   where $\sigma()$ is some non-linear activation function.

   (a) In the term above, look at what happens when example n is of class 1? And then what happens when example n is of class 2? Show that we can define a target label $t_{n,12}$ (to be read $1, 2$, one-two, not twelve!) such that this expression looks more like what we're used to. (0.5 pt)

   (b) Explain how $\vec{w}_{21}$ depends on $\vec{w}_{12}$. How does $t_{n,21}$ depend on $t_{n,12}$? (0.25 pt)

   (c) Generalize the term of equation 1 to $K > 2$ classes. In particular, define our target label $t_{n,kk'}$. (0.75 pt)

2. In the $K > 2$ case, how many parameters are there in the model (what is the cardinal of Θ)? You may count "naively" and then count only independent parameters (making use of the relationship(s) found earlier). In the rest of the exercise, we may ignore this dependence and encode the parameters in a rather simple matrix, of simple shape. (0.5 pt)

3. We choose a Least-Squared Error kind of loss function, i.e. we want to define the loss J (or $\mathcal{L}$, as you prefer) as the appropriate sum over examples and classes, of the square of the term found in the previous question (generalization of equation 1 to $K > 2$ classes). Write down explicitly the full cost function of our model, $J(\Theta, X)$. Be careful to not forget some sums ($\sum_{\dots}$)! (0.5 pt)

4. (This question depends on question 3) The hyperbolic tangent function is defined by: $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. It goes from -1 at $x \sim -\infty$ to $+1$ at $x \sim +\infty$. Its derivative is (for a generic smooth function u): $\partial_x \tanh(u(x)) = \frac{4u'(x)}{(\cosh(u(x)))^2}$, where $\cosh(x)$ is the hyperbolic cosine, $\cosh(x) = \frac{e^x + e^{-x}}{2}$. We know that $\cosh(x) \geq 1, \forall x$. We decide to use $\sigma(.) = \tanh(.)$, since $\tanh(.)$ nicely interpolates between -1 and $+1$.

   (a) Derive the elementary step of the Gradient descent algorithm for this $J(\Theta, X)$. If you cannot compute it, or you aren't sure of the result, in the rest of the exercise, you may simply write ∇J when it's convenient (and not explicit what it is). Explicit all indices. (1 pt)

   (b) Write the GD step in Matrix form, and give the shapes of all matrices or vectors involved, to make sure your computation is a legal computation. (0.5 pt)

5. We are interested in what will happen if we do updates taking points one by one (like in SGD or Online Perceptron). In particular, what happens to plane $\vec{w}_{kk'}$ when we visit a point of class k'', with $k'' \neq k$, and $k'' \neq k'$? (0.5 pt)

* We can now suppose the model has been learned. You may define some $f_\Theta(\vec{x})$ function, to lighten your notations. From now on, we assume the parameters have been learned.

6. For a test point $\vec{x}^{test}$ (for simplicity, $\vec{x}^{test} = \vec{x}$), what are the predictions of the hyperplanes $w_{kk'}$? How many do we have? We decide to take the max value. Write the corresponding `predict` function formally (mathematically). (0.75 point)

7. Choose a performance metric for this task, and write it in mathematical form for our model. (0.25 point)

8. Specify the list of our hyper-parameters (the main ones at least). (0.5 point)

Bonus Write down the `fit` and `predict` function in pseudo-code, sklearn-style. You should assume the data is given, but you should initialize the parameters and hyper-params yourself (as in real code). (up to 1 point? I don't know)

4 Identifying the task, the appropriate classes of algos, and data structures/features (5 points)

In this exercise, we ask you to give a general idea about the problem, not specify the details in mathematical forms. You can use words and simple sketches ("des petits dessins") to explain your ideas.

For each of the following projects, answer these questions:

(a) What is the goal?

(b) What kind of task should we probably do (supervised or not, and in each case, which sub-category)?

(c) What is the data structure? (you can and should often make some the assumptions you want)

(d) What algorithm seems well suited?

(e) Are there some particular things we should worry about (hyper-parameter choice, performance metric, unbalanced data set, etc).

Apart from (e), most of the questions are answered in a single line ! In cases (3) and (4), there is an additional question, which completes the question (e). For question (e), each time, a couple of line ($\approx$ 5 lines) should be enough discussion. Be concise.

1. (1 pt) We have a data set of mushrooms properties, one of which is a binary attribute, "edible/poisonous". Other attributes are often categorical: cap-shape, cap-surface, cap-color, odor, habitat, etc. There are many more edible mushrooms than poisonous ones. We are **interested in not dying from poison**.

2. (1 pt) In the game of pokemon, each pokemon has some particular abilities, that can be very efficient against some other kind of pokemons. A fight is performed with a fixed set of pokemons on each side of the fight. Here we have a simple case where there is only one pokemon on each side. We have a dataset of thousands of pokemon fights. In each case, we know some features of each pokemon: Name, Hit Points, Attack, Defense, Speed, type (water, fire, psy, ground, etc). The fight is always won by a team, team 1 or team 2. A given pokemon type (for instance name=pikachu) can appear in multiple fights (but not always with the exact same number of Hit Poins, attack, etc).
   We are **interested in predicting who may win a given fight**.

3. (1.5 pt) World temperatures. We have records of temperature in cities all over the world, going back $\sim$ 100 years. For most cities, the record is done once per month.
   We are **interested in future temperatures**. We have the time of recording, the temperature, and the latitude and longitude of the city. What can we do?
   Point (e): In this project, can you think of a smart feature map? (remember that seasons are periodic over the year, and seasonal variations are stronger than the global warming trend).

4. (1.5 pt) Footballers positions. We have a dataset of thousands of professional football players' data. For each player, there are many attributes. Some are categorical: name, weak foot, preferred foot. Some are quantitative: weight, height, age, acceleration. Some are given by experts of the field, and expressed as rates (numbers between 0 and 100): Speed, Stamina, Strength, Balance, Agility, Jumping, Heading.
   We are **interested in predicting their playing position** (center, left, right, and front, middle, back, goal).
   Point (e): There is an additional problem to this task. Playing positions labels are not standardized across the world. There is a total of 27 possible label values in our dataset, which is much more than the 9-12 positions that are commonly considered in expert football. We are interested in reducing this diversity of playing positions to a reasonable number. How would you do this reduction ?