Examen cours AMAL (Advanced Machine Learning) - Masters DAC et M2A – Sorbonne Université

8/02/2022 – Documents autorisés - Durée 2h

MCQ: answer Y/N

N 1. The double descent phenomenon in deep NNs characterizes a gradient acceleration technique
2. Double descent has been characterized for very large neural networks
3. Transposed convolutions are often used for upsampling images
4. In ResNets, skip connections have been introduced to improve the stability of classical NNs
5. The gating mechanism in GRUs makes use of a form of skip connection
6. The skip gram model is a language model
7. The skip gram model is trained to classify input words
8. The skip gram model learns semantic similarities
9. The transformers are language models
10. Attention models learn combinations of their inputs
11. In transformers, self attention is a sequence to sequence operation
12. Transformer are encoder-decoder architectures
13. For SVMs, support vectors fully define the decision frontier
14. The dual formulation of SVMs is mainly used for linear kernels
15. Gaussian processes are trained to predict conditional distributions over functions
16. Gaussian processes are fully defined by a mean function and a covariance function
17. Gaussian processes are Bayesian methods
18. Neural processes are kernel methods
19. Neural processes allow to predict a mean value and an uncertainty on this mean value
20. Neural processes make use of a series of datasets for training

Exercise - Domain generalization

We consider a training set D_{Tr} consisting of S domains $D_{Tr} = \{D_1, \dots, D_S]$, each domain is characterized by a dataset $D_s = \{(x_i^s, y_i^s); i = 1 \dots N_s\}$ containing data drawn from some probability distribution; the distributions over the domains characterize a similar phenomenon, but are different – for example the data have been captured in different conditions in each domain. The goal of domain generalization is to train a model with weights θ that generalizes well on test datasets.

Let $l((x, y); \theta)$ the loss of model θ evaluated at (x, y). The classical approach is to shuffle the data from the different domains, ignoring their specificities and train according to the empirical risk minimization (ERM) principle. The ERM loss is:

$$\mathcal{L}_{ERM} = E_{D \sim D_{Tr}} E_{(x,y) \sim D}[l(x, y); \theta]$$

With D_{Tr} the distribution on the domains. This naïve approach fails to generalize on new domains. The objective of the exercise is to examine alternative approaches.

1. Pitfalls of ERM for domain generalization

We will first examine a simple example of binary classification with a linear classifier. Let $x \in \{0,1\}^4, y \in \{0,1\}$. We consider three domains, D_1, D_2, D_3, the training set is $\{D_1, D_2\}$ the test set is D_3. The setup is illustrated on the figure below.

	$p(x_1) = 0.5$	$p(x_2) = 0.4$	$p(x_3) = 0.1$
D_1	0 0 0 0	1 1 0 0	1 0 0 0
D_2	0 0 0 0	1 0 1 0	1 0 0 0
D_3	0 0 0 0	1 0 0 1	1 0 0 0
	$p(y = 1 \mid x_1) = 0$	$p(y = 1 \mid x_2) = 1$	$p(y = 1 \mid x_3) = 0.3$

The three domains (rows) consist of 3 types of inputs (columns) x_1, x_2, x_3. Each type respectively makes for 50%, 40%, 10% of each domain (indicated in the first row). For x_1 label is always 0, for x_2 it is always 1 and for x_3 it is 1 with 30% (indicated in the last row). A classifier trained on (D_1, D_2) performs at 97% accuracy on train and 57% on test D_3.

What is the accuracy (probability of correct classification) of the optimal classifier if for the decision, one considers only the first feature f_1 for each input $x = (f_1, f_2, f_3, f_4)$, for any of the three domain (note that the accuracy is the same for all the domains if one considers only f_1)? What can you conclude on the non-optimality of the ERM principle for this example? Indication: the optimal decision is decide $y = 0$ *if* $f_1 = 0$, and $y = 1$ *if* $f_1 = 1$. The accuracy of this decision is calculated as $p(y = 0, f_1 = 0) + p(y = 1, f_1 = 1)$

2. Meta-learning

One will consider an alternative solution to ERM, based on meta-learning. Let us split the training set D_{Tr} into two sets of domains $D_{Tr} = D_A \cup D_B$. D_A and D_B each contains domains D_i from D_{Tr}. The objective of meta-learning is to train on the sets in D_A so as to generalize well on the sets in D_B. The losses on the two sets of domains are respectively denoted:

$$\mathcal{L}_{D_A}(\theta) = E_{D_A}[l(x, y); \theta] = \frac{1}{|N_{D_A}|} \sum_{i=1}^{N_{D_A}} \frac{1}{N_i} \sum_{j=1}^{N_i} l(x_j^i, y_j^i; \theta)$$

$$\mathcal{L}_{D_B}(\theta) = E_{D_B}[l(x, y); \theta] = \frac{1}{|N_{D_B}|} \sum_{i=1}^{N_{D_B}} \frac{1}{N_i} \sum_{j=1}^{N_i} l(x_j^i, y_j^i; \theta)$$

With N_{D_A} the number of sets in D_A, N_i the number of data elements (x, y) in set D_i, x_j^i the j^{th} element in set D_i, the inside summations are respectively over each set D_i in D_A and D_B.

The meta loss is defined as:

$$\mathcal{L}_{M-L}(\theta) = \mathcal{L}_{D_A}(\theta) + \mathcal{L}_{D_B}(\theta - \alpha \nabla \mathcal{L}_{D_A}(\theta)))$$

with $\nabla \mathcal{L}_{D_A}(\theta)$ the gradient vector of $\mathcal{L}_{D_A}(\theta)$ w.r.t. θ. i.e. one wants that the modifications on θ minimize $\mathcal{L}_{D_A}(\theta)$ while the gradient step also minimizes $\mathcal{L}_{D_B}(\theta')$ with $\theta' = \theta - \alpha \nabla \mathcal{L}_{D_A}(\theta)$.

2.1 Give the expression of the order 1 Taylor expansion of $\mathcal{L}_{D_B}(\theta')$ around θ. The formula for the Taylor expansion for a vectorial function is given at the end of the text.

2.2 Show that neglecting the residual terms of the expansion leads to the following expression:

$$\mathcal{L}_{M-L}(\theta) = \mathcal{L}_{D_A}(\theta) + \mathcal{L}_{D_B}(\theta) - \alpha(\nabla \mathcal{L}_{D_A}(\theta) . \nabla \mathcal{L}_{D_B}(\theta))$$

with $\nabla \mathcal{L}_{D_A}(\theta) . \nabla \mathcal{L}_{D_B}(\theta)$ the dot product of the two gradient vectors.

2.3 Interpret this expression of the loss function. What is the usefulness of maximizing $\nabla\mathcal{L}_{D_A}(\theta).\nabla\mathcal{L}_{D_B}(\theta)$?

3. Gradient alignment

Using this idea of gradient alignment, we consider a new loss criterion.

Let us denote $G_i = E_{D_i}[\frac{\partial l(x,y;\theta)}{\partial\theta}]$, the gradient computed on dataset D_i. The new loss is:

$$\mathcal{L} = \mathcal{L}_{ERM}(D_{Tr};\theta) - \gamma\frac{2}{N_{D_{Tr}}(N_{D_{Tr}}-1)}\sum_{i,j\in D_{Tr}, i\neq j} G_i.G_j$$

with $G_i.G_j$ the dot product of the two gradient vectors and $N_{D_{Tr}}$ is the number of sets in D_{Tr}. Algorithm 2 below is a direct gradient optimization of $\mathcal{L}$, and Algorithm 1 approximates this optimization.

```
Algorithm 1 - Approximation
Initialize $\theta$
$\tilde{\theta} = \theta$
for iterations=1,2,... do
  for $D_{p[i]} \in permutation\{D_1, \dots, D_S\}$ do
    $g_i = E_{D_{p[i]}}[\frac{\partial l(x,y;\tilde{\theta})}{\partial\tilde{\theta}}]$ gradient w.r.t. $\tilde{\theta}$
    Update: $\tilde{\theta} = \tilde{\theta} - \alpha g_i$
  endfor

  Update $\theta = \theta - \epsilon(\theta - \tilde{\theta})$
endfor
```

```
Algorithm 2 – Direct optimization
Initialize $\theta$
for iterations=1,2,... do
  for $D_i \in permutation\{D_1, \dots, D_S\}$ do
    $G_i = E_{D_i}[\frac{\partial l(x,y;\theta)}{\partial\theta}]$ gradient w.r.t. $\theta$
  endfor
  $\bar{G} = \frac{1}{N_{Tr}}\sum_{s=1}^{N_{Tr}} G_s$, $\hat{G} = \frac{2}{N_{Tr}(N_{Tr}-1)}\sum_{i,j\in D_{Tr}, i\neq j} G_i.G_j$

  Update $\theta = \theta - \epsilon(\bar{G} - \gamma\frac{\partial\hat{G}}{\partial\theta})$
endfor
```

Both algorithms have an outside loop (the first **for**) and an inside loop (the second **for**). "**for** $D_i \in permutation\{D_1, \dots, D_S\}$ " indicates that one of the domains is sampled from $\{D_1, \dots, D_S\}$ so that in the inner loop, all the domains will be selected in turn. $p[i]$ denotes the index of the selected set Dat the i^{th}iteration of the inner loop.

3.1 What makes Algorithm 2 computationally prohibitive? Why is Algorithm 1 far less computationaly demanding than Algorithm 2?

3.2 Let us consider two domains D_1, D_2, give the expression of the outer loop update (θ) for each algorithm.

3.3 Let us analyze one iteration of the outer loop. We will show that the directions of the update are the same for the two algorithms. We will consider in the following the inner loop of algorithm 1. Let us first introduce some notations. **In the following the index, i corresponds to the i^{th} iteration in the inner loop** and not to the domain index.

- $L_i = L(D_{p[i]};\tilde{\theta}_i)$ is the loss at the i^{th} iteration, where $p[i]$ indicates the domain selected at the i^{th} iteration.
- $g_i = \nabla_{\tilde{\theta}_i} L_i = L_i'(\tilde{\theta}_i)$ is the gradient at the i^{th} iteration (on $D_{p[i]}$)
- The update rule at step i is $\tilde{\theta}_{i+1} = \tilde{\theta}_i - \alpha g_i$
- $\theta_1$is the parameter value at the start of the inner loop
- $g_{i,1} = L_i'(\theta_1)$ is the gradient w.r.t. θcomputed on the domain $D_{p[i]}$ at θ_1
- $h_{i,1} = L_i''(\theta_1)$ is the hessian computed on domain $D_{p[i]}$ at θ_1

Note: the gradients are vectors, the hessian is a matrix, all of the appropriate size.

3.3.1 Using a $\overset{1}{2}$nd order Taylor expansion of $L_i'(\bar{\theta}_i)$ around θ_1, show the following identity:

$$g_i = L_i'(\theta_1) + L_i''(\theta_1)(\bar{\theta}_i - \theta_1) + O(\alpha^2)$$

Indication: use $L_i'(\bar{\theta}_i) = L_i'(\theta_1 + (\bar{\theta}_i - \theta_1))$

Show then:

$$g_i = g_{i,1} - \overset{\alpha}{h_{i,1}} \sum_{j=1}^{i-1} g_j + O(\alpha^2)$$

3.3.2 Using the order 1 expansion around θ_1, $g_j = g_{j,1} + O(\alpha)$show that

$$g_i = g_{i,1} - \alpha h_{i,1} \sum_{j=1}^{i-1} g_{j,1} + O(\alpha^2)$$

3.3.3 Let us now consider an example with two domains D_1, D_2 and let us suppose that in the inner loop, they are selected in this order D_1 then D_2. In the two successive iterations in the inner loop, one gets:

$$L_1 = L(D_1; \theta_1); g_1 = L_1'(\theta_1)\,; \bar{\theta}_2 = \theta_1 - \alpha g_1$$
$$L_2 = L(D_2; \bar{\theta}_2); g_2 = L_2'(\bar{\theta}_2)\,; \bar{\theta}_3 = \bar{\theta}_2 - \alpha g_2$$

Show that the update for the outer loop writes $\Delta\theta = \theta_1 - \bar{\theta}_3 = \alpha(g_1 + g_2)$

3.3.4 Using the above results show that

$$g_1 + g_2 = g_{1,1} + g_{2,1} - \alpha h_{2,1} g_{1,1}$$

3.3.5 Note that in this equation, $g_{1,1} = G_1(\theta_1), g_{2,1} = G_2(\theta_1), h_{2,1} = \nabla G_2(\theta_1)$, (a word of caution here: with lowercase letters (e.g. g_1), indexes correspond to inner loop iteration, while for capital letters (e.g. G_1), it corresponds to the domain). What would be the value of $g_1, g_2, h_{2,1}$ if the domains had been considered in the order D_2 then D_1 in the inner loop?

3.3.6 Let us now consider the expectation of $g_1 + g_2$ w.r.t. the sampling on the domains (the **for** loop in the inner loop). In the case of two datasets, this expectation is simply the mean of $g_1 + g_2$ computed over the two possible samplings, (D_1 then D_2) and (D_2 then D_1).Show that this expectation writes as ($H_1(\theta_1)$ denotes the hessian of the loss function on D_1 at θ_1:

$$E_{permutations}[g_1 + g_2] = G_1(\theta_1) + G_2(\theta_1) - \frac{\alpha}{2}\big(H_1(\theta_1)G_2(\theta_1) + H_2(\theta_1)G_1(\theta_1)\big) + O(\alpha^2)$$

$$E_{permutations}[\Delta\theta] = 2\alpha\bar{G} - \frac{\alpha^2}{2}\frac{\{\partial G_1 . G_2\}}{\partial \theta_1}$$

3.3.7 Compare with the gradient direction in Algorithm 2 and conclude.

Taylor expansion

Let $f: \mathbb{R}^p \rightarrow \mathbb{R}$ a twice differentiable function, the Taylor-Young expansion of order 2 around point a is: $f(a+h) = f(a) + \nabla f(a).h + \frac{1}{2}h^T H(a)h + O(\|h\|^2)$ with $+\nabla f(a)$ the gradient of $f(a)$ and $H(a)$ its hessian matrix

The Taylor-Young expansion of order 1 is: $f(a+h) = f(a) + \nabla f(a).h + O(h)$