CS 4644/7643: Deep Learning
Spring 2023
HW1 Solutions

Giulia Paggini

May 11, 2023

1 Collaborators

- Annalisa Barbara
- Pierluigi Mancinelli
- Susanna Paoli

2 Optimization

2.1

Prove that the gradient is orthogonal to the tangent of a curve $r(t)$ passing through t_0.
Consider $f(x, y, z) = c$, a point (x_0, y_0, z_0) that lies on the surface and any parametrical curve passing through it at $t = t_0$,defined by:

$$x = x(t) \;\; y = y(t) \;\; z = z(t)$$

then

$$f(x(t), y(t), z(t)) = c$$

where $x(t) = (x(t), y(t), z(t))$ Given that, we can differentiate both sides using the chain rule:

$$f_x(x, y, z)x' + f_y(x, y, z)y' + f_z(x, y, z)z' = 0$$

but this is the dot product between a tangent generic vector (x', y', z') and the gradient of f:

$$\nabla f(x', y', z') * x'(t) = 0$$

this means that the gradient is orthogonal to x' but it is also tangent to the curve, arbitrarily picked on the surface, therefore the gradient is perpendicular to the surface too.
This is in general true also in higher dimensions since f assumes a constant value in a surface and its derivative is 0. The use of gradient in deep learning is fundamental for optimization purposes, in particular given a loss function to minimize, the gradient always points the steepest direction to follow, reaching a point of convergence in the fastest way possible.

2.2

Since g has a local minimum at w^t, for small $t \neq 0$, we have that:

$$g(w^t + t) - g(w^t) \geq 0$$

- if $t > 0$:
$$\frac{g(w^t + t) - g(w^t)}{t} \geq 0$$
- if $t < 0$:
$$\frac{g(w^t + t) - g(w^t)}{t} \leq 0$$

but we know that g' exists therefore the right and left limits must be equal to $g'(w^t)$. Since $g'(w^t) \geq 0$ and $g'(w^t) \leq 0$ it must be that $g'(w^t) = 0$ so the gradient computed at w^t must be zero.
For the converse is enough to consider a counterexample with $g(x) = x^3$.

$$\nabla g(x^*) = 3x^2$$

$$\nabla g(x^*) = 0 \;\; 3x^2 = 0 \;\; x^* = 0$$

$$\nabla g(x^*) = 0 \;\; 3x^2 = 0 \;\; x^* = 0$$

$$\nabla^2 g(x^*) = 6x \quad x^* = 0 \geq 0$$

x^* is saddle point for the function above and not a minimum.

2.3

If g is a convex function then it must be that:

$$g(\lambda y + (1-\lambda)w^*) \leq \lambda g(y) + (1-\lambda)g(w^*)$$

$$g(w^* + \lambda(y - w^*)) \leq g(w^*) + \lambda(g(y) - g(w^*))$$

$$g(y) - g(w^*) \geq \frac{g(w^* + \lambda(y - w^*)) - g(w^*)}{\lambda}$$

As $\lambda \to 0:$
$g(y) - g(w^*) \geq \nabla g^T(w^*)(y - w^*)$ but we know that $\nabla g^T(w^*) = 0$ therefore $g(y) \geq g(w^*)$. Thus for any value different from w^*, g assumes higher values, so w^* is a global minimum.

2.4

First of all, let's write down the generic Jacobian of the softmax function:

$$J_{softmax} = \begin{bmatrix} \frac{\partial S_1}{\partial z_1} & ... & ... & \frac{\partial S_1}{\partial z_n} \\ \frac{\partial S_2}{\partial z_1} & \frac{\partial S_2}{\partial z_2} & ... & \frac{\partial S_2}{\partial z_n} \\ ... & ... & ... & ... \\ \frac{\partial S_n}{\partial z_1} & ... & ... & \frac{\partial S_n}{\partial z_n} \end{bmatrix}$$

Now consider for example $i = 1$ and $k = 3$.

Case $\quad i = j = 1$

$$S_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

$$\frac{\partial S_1}{\partial z_1} = \frac{e^{z_1}(e^{z_1} + e^{z_2} + e^{z_3}) - e^{z_1}e^{z_1}}{(e^{z_1} + e^{z_2} + e^{z_3})^2}$$

$$= \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}} \frac{e^{z_2} + e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

$$= s_i \frac{\sum_{k \neq i} e^{z_k}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

$$= s_1(1 - s_1)$$

Case $\quad i = 1 \quad$ and $\quad j = 2$

$$\frac{\partial S_1}{\partial z_2} = \frac{e^{z_1}(-e^{z_2})}{(e^{z_1} + e^{z_2} + e^{z_3})^2}$$

$$= \frac{-e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}} \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

$$= -s_1 s_2$$

analogously you obtain the same results for $i = 1$ and $j = 3$.
More in general:

$$J_{softmax} = \begin{bmatrix} s_i(1 - s_i) & -s_i s_j & ... & -s_i s_j \\ -s_i s_j & s_i(1 - s_i) & ... & -s_i s_j \\ ... & ... & ... & ... \\ -s_i s_j & ... & ... & s_i(1 - s_i) \end{bmatrix}$$

where i is the number of the row and j indicates the column.

2.5

First of all we write down the Lagrangian function:

$$
\begin{gathered}
L(x,y,\lambda) = -x^T y - H(y) + \lambda(1^T y - 1) \\
\nabla_y L(x,y,\lambda) = -x_i + log(y_i) + 1 + \lambda = 0 \\
log(y_i) = x_i - 1 - \lambda \\
y_i^* = e^{(x_i - 1 - \lambda)}
\end{gathered}
$$

then we substitute back in the constraint:

$$
\begin{gathered}
\sum_i e^{(x_i - 1 - \lambda *)} = 1 \\
\lambda^* = log \sum_i e^{(x_i - 1)} \\
y_i^* = e^{x_i - 1 - log(\sum_i e^{(x_i - 1)})} \\
= e^{x_i - 1} \frac{1}{\sum_i (e^{x_i - 1})} \\
= \frac{e^{x_i}}{e} \frac{e}{\sum_i e^{x_i}} \\
= \frac{e^{x_i}}{\sum_i e^{x_i}}
\end{gathered}
$$

The softmax function we have just obtained is widely used in deep learning applications, in particular it is applied on the last layer of a neural network for classification tasks in order to convert scores in probabilities, to make them interpretable. Scores are not practical to identify the prediction of the class because there is no bounded value as a reference.

3 Directed Acyclic Graphs

3.1

Prove that if G is a DAG, then has a topological ordering.

By induction
As usual, we will go through the base case, then we will make an assumption about a case with n nodes and see whether it is true for the $n + 1$ case. The base case with $n = 1$ the topological order is obvious, there is only one node. Our hypothesis is that for a graph with $\leq n$, G has a topological ordering. Then consider a DAG with $n+1$ nodes. The idea is to find a vector v that has no incoming edges. Then G -{v} is still a DAG, there are also no cycles and according to our hypothesis has a topological ordering. By placing v at the beginning and append G -{v} in topological order as v is the root of our DAG.

3.2

Prove that if G has a topological order, then G is a DAG.

By contradiction Let's suppose that a graph G has a topological ordering $v_1, v_2, ..., v_n$ and a directed cycle called C.
Denote with v_i the smallest index involved in C and v_j the node before it, then (v_j, v_i) is an edge. But in this case $i < j$ but since (v_j, v_i) is an edge, $v_1, v_2, ..., v_n$ is a topological order and $j < i$ which is indeed a contradiction we conclude that a graph G has no cycle, so it's a Directed Acyclic Graph.

4 Paper Review

4.1

The proposed paper starts with an analogy between animals, that have specific capabilities since their birth, and Neural Networks that well perform even if their weight parameters are randomly sampled. The leitmotif throughout the paper is the one of reducing the emphasis on weights, by giving more relevance to the architecture per se. In Weight Agnostic NN, first of all a set of minimal architectures is created, then by using shared weights values these are evaluated one by one, then ranked by performance and complexity, finally other architectures are created as a variation of the best ones just found.
As strengths I would mention the time saving in training the shared weights, as well as the memory cost which is significantly reduced compared to a classical NN, but also they can be used for algorithms that do not require the computation of the gradient. The randomness of the weights also allows for using several copies of the same WANN with different values, so that an ensemble can be created to score better results than a single model.
As a downside, I would cite that the method for finding optimal weights indeed matters, and sometimes the technique of shared weights may not be the most performing one as in the case of MNIST. Traditional neural networks are more task-specific and with specific architectures can really outperform WANNs, like CNNs do with an accuracy of 99% on the same dataset.

4.2

Personally I found this paper promising and at the same astonishing for the results obtained using WANNs on well-know problems, either in reinforcement learning but also in classification tasks. My personal takeaway from this paper is that with this revolutionary method the training of a NN will be much more lighter, and the structures derived by WANNs serve as a starting point for better designs.

The fact that the authors open sourced the code for WANN is of course an encouragement to progress in this way in the next future and in particular an attempt to save time and reduce the computational costs when training complex neural architectures. In addition to that, a quick fine-tune learning could be useful in the field of continual learning or in general for models that require a continuous training and update during a certain time span.

5 Implement and train a Network of MNIST

- After having performed all the trainings with different learning rates as proposed, we can observe that the two highest levels accuracy are reached, respectively, with a learning rate equals to 1 and 0.1. In principle, a larger learning rate could be a better option in order to reduce the time needed for the algorithm to converge but at the same time, it could causes drastic updates during the optimization steps. Instead, as a more reasonable choice, I would suggest to pick lr=0.1 since it scores a pretty good level of accuracy, while a smaller rate could prevent a possible overshooting. The other two learning rates show a lower accuracy perhaps, during the optimization, the gradient gets stuck in a point of local minimum.

- Looking at the parameters above for regularization, we can conclude that the best one for our model is reg=0.0001, since it reaches the highest accuracy among the others and helps the model reducing a possible overfitting. The worst is definitely reg=1 as we can easily notice from the loss and accuracy plots: a too heavy penalization results in a poor level of accuracy.

- For this final section, after some trail and error, I decided to set: lr = 0.6, so that it is neither too large, nor too small, reg = 0.0005, momentum = 0.85 and a slightly higher number of epochs, namely 20.

  The final accuracy reached by the model is around 96.50% which is reasonable for the given dataset. In principle, as stated in the WANN paper, one could score even 99% of accuracy but in this case, given the fact that the plots of the train and valid curves are so close, I preferred to use this hyperparameters to avoid overfitting, and the plots to evidently crossed each other.