Vector Calculus for Machine Learning 1

Rob Hesselink
&
Sharvaree Vadgama

September 2021

1 Introduction

Hi! This document discusses some of the mathematics necessary to get the most out of the course 'Machine Learning 1', for the MSc Artificial Intelligence at the University of Amsterdam. It is intended for those students that feel they've lost sight on what derivatives are, how to take derivatives with respect to vectors or matrices and why they might need to differentiate anything to begin with.

We will start with answering 'why' and then continue to 'how' in some of its many forms, as functions come in all shapes and sizes. This document was written to accompany lectures given in September 2021, but it should be self-contained. Without further ado, let's begin!

2 Why would I differentiate a function?

The answer to this is quite simple: *optimisation*. While there are many ways to find the optima of a given function, it so happens that computing the derivative (or gradient) is a convenient way to find the optima.

Any supervised learning problem, given dataset $X = \{x_1, ..., x_N\}$, targets $Y = \{y_1, ..., y_N\}$, a model f_θ and loss function L, can be expressed as

$$\arg\min_\theta \sum_{i=1}^{N} L(f_\theta(x_i), y_i) \tag{1}$$

This is evidently optimisation, where we try to find the ideal set of parameters for this specific problem. As some of you might recall, a necessary condition for an optimum for a function f—assuming that it's differentiable—is that its derivative $\frac{df}{dx} = 0$.

Some days $\frac{df}{dx} = 0$ will have a closed-form solution, which implies that we can analytically solve the optimisation problem. Other times, we'll be out of luck and will have to resort to approximate methods such as *gradient descent*. And to make matters worse, *local optima* exist. It might be very difficult to determine whether a given optimum is locally optimal, or globally optimal. Figure 1 illustrates some of these.

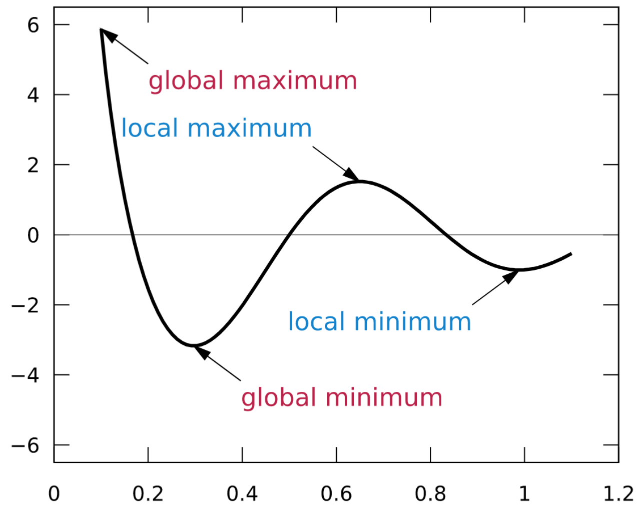

Figure 1: Local and global maxima and minima for $\cos(3\pi x)/x, 0.1 \leq x \leq 1.1$. Taken from Wikipedia.

Of course, not all functions have global optima, as they might continue to grow forever towards infinity. And periodic functions will have an infinite number of them.

3 Scalar derivatives

Let us begin differentiating using good old scalar functions and variables. One such example is $f(x) = x^2$. You might remember that $\frac{df}{dx} = 2x$ and if so, good for you! But why is this so?

Well, the derivative measures the response of a function's output, when changing the input infinitesimally. As such, the definition of a derivative is

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{2}$$

If we plug our example into this definition, we will get

$$\begin{aligned}\lim_{h\to 0}\frac{(x+h)^2-x^2}{h} &= \lim_{h\to 0}\frac{x^2+2hx+h^2-x^2}{h}\\ &= \lim_{h\to 0}\frac{2hx+h^2}{h}\\ &= 2x.\end{aligned}$$

We took the limit in the last line, which was safe to do in this particular case. We can do the same trick for many functions and get some standard results, but it's often best just to remember them. Some of my personal favourites are:

- $\frac{d}{dx}x^n = nx^{n-1}$
- $\frac{d}{dx}\ln(x) = 1/x$
- $\frac{d}{dx}e^{-x} = -e^{-x}$
- $\frac{d}{dx}3 = 0$

The limit of more complicated functions may not always be well-defined. For example, a piece-wise function has a well-defined derivative only if the limits from both sides agree on the value of the derivative. However, in this course we will generally deal with 'nice' functions.

There are two important rules that we need to go over, that will greatly simplify any differentiation you carry out.

3.1 Product rule

The derivative of a product is the sum of the individual derivatives:

$$\frac{d}{dx}(u\cdot v) = \frac{du}{dx}\cdot v + u\cdot\frac{dv}{dx} \tag{3}$$

For example:

$$\begin{aligned}\frac{d}{dx}\left[x^2\ln(x)\right] &= 2x\ln(x)+\frac{x^2}{x}\\ &= 2x\ln(x)+x\end{aligned}$$

3.2 Chain rule

This one might be even more important. If the function you're differentiating consists of functions-of-functions, this rule allows you to break the computation into multiple parts.

Whenever f(x) = f(g(x)),

$$\frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx} \tag{4}$$

For example, given $f(x) = e^{\sin(x)}$, set $u = \sin(x)$:

$$\begin{aligned}\frac{df}{dx} &= \frac{df}{du}\frac{du}{dx} \\ &= e^u \cos(x) \\ &= e^{\sin(x)} \cos(x) \\ &= f(x)\cos(x)\end{aligned}$$

4 Vector calculus

To tell you a secret, I don't know how to take vector derivatives. **All I can do is take scalar derivatives with of components of vectors and infer the full shape afterwards**. This is exactly the recipe that I will follow in the following examples. Of course, you can choose to memorise some common vector/matrix derivatives, but if you ever need to figure them out yourself, this method will always work.

A quick overview:

1. Transform into componentwise notation
2. Take componentwise derivative, using scalar derivation
3. Infer the shape of the output using agreed upon shape conventions

4.1 Componentwise notation

Let's remind ourselves what the dot product or matrix multiplication looks like. Generally, it is a sum over components, but you do need to be careful about using the right indices. Below follow some examples:

$$\begin{aligned}
\mathbf{v}^T\mathbf{v} &\rightarrow \sum_i v_i v_i = x \\
\mathbf{A}\mathbf{v} &\rightarrow \sum_j A_{ij} v_j = x_i \\
\mathbf{v}^T\mathbf{A} &\rightarrow \sum_i A_{ij} v_i = x_j \\
\mathbf{A}\mathbf{B} &\rightarrow \sum_j A_{ij} B_{jk} = X_{ik} \\
\mathbf{v}^T\mathbf{A}\mathbf{v} &\rightarrow \sum_{i,j} v_i A_{ij} v_j = x
\end{aligned}$$

Perhaps it's good to note that if you're expecting the output to be a vector, there should only be one index. Similarly, if it's a matrix, you would expect two indices.

Another pitfall is that, when using componentwise notation, there is some loss of information about the shape of the original object. For example, we could write

$$\mathbf{A}\mathbf{v} \rightarrow \sum_j A_{ij} v_j \tag{5}$$

but at the same time, with the same conventions, we could also write

$$\mathbf{v}^T\mathbf{A}^T \rightarrow \sum_j A_{ij} v_j \tag{6}$$

These are identical, even though the former yields a column vector and the latter a row vector. Fortunately, any ambiguity in the shapes can be resolved by the shape conventions that we introduce in section 4.3.

4.2 Componentwise differentiation

Let's start with an example: $\mathbf{f}(\mathbf{v}) = \mathbf{A}\mathbf{v}$. Here $A \in \mathbb{R}^{m\times n}, v \in \mathbb{R}^n$ and $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$. I would like to compute $\frac{d\mathbf{f}}{d\mathbf{v}}$, so I'll write the function in component-form:

$$f_i = \sum_j A_{ij} v_j \tag{7}$$

Then I'll take the derivative[1] with respect to a **different component**:

[1] Note that I'm suddenly swapping from a normal d to a slanted ∂. The latter indicates a partial derivative, where I keep other variables fixed and only vary one.

$$\frac{\partial f_i}{\partial v_k} = \frac{\partial}{\partial v_k} \sum_j A_{ij} v_j \tag{8}$$

$$= \sum_j \frac{\partial}{\partial v_k} A_{ij} v_j \tag{9}$$

It seems that this expression is only non-zero whenever $k = j$. Fortunately, there exists a nice mathematical object to express precisely this: the Kronecker delta. This object is defined as:

$$\delta_{ij} = \begin{cases} 1 & \text{if i = j} \\ 0 & \text{else} \end{cases} \tag{10}$$

In our previous expression the summation over the index j makes sure that this happens exactly once. This means that our answer becomes

$$\frac{\partial f_i}{\partial v_k} = \sum_j A_{ij} \delta_{jk} \tag{11}$$

$$= A_{ik} \tag{12}$$

The Kronecker delta 'filtered' a single component from the summation over j, multiplying all others by zero.

4.3 Shapes

Now for inferring the shape, we must address some conventions first. After all, the full answer might be $\mathbf{A}$ or $\mathbf{A^T}$. Different works use different conventions, but we will stick with those found in the book 'Mathematics for Machine Learning'. This means that the default vector will be a column vector, e.g. $\mathbf{v} \in \mathbb{R}^n$ and the derivative of a scalar with respect to a column vector is a row vector:

$$\nabla_{\mathbf{v}} f = \frac{df}{d\mathbf{v}} = \left[\frac{\partial f}{\partial v_1}, ..., \frac{\partial f}{\partial v_n} \right] \in \mathbb{R}^{1 \times n} \tag{13}$$

Note that some works will say the gradient and the derivative are each others' transpose. For this course, there will be no such distinction and they will be used interchangeably. The previous result can be extended to the case of a vector-valued function $\mathbf{f}$, like so:

$$\nabla_{\mathbf{v}}\mathbf{f} = \frac{d\mathbf{f}}{d\mathbf{v}} = \begin{bmatrix} \frac{\partial f_1}{\partial v_1} & \cdots & \frac{\partial f_1}{\partial v_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial v_1} & \cdots & \frac{\partial f_m}{\partial v_n} \end{bmatrix} \in \mathbb{R}^{m\times n} \tag{14}$$

As such, we have reached the final answer of our derivation and can now say that

$$\frac{\partial \mathbf{f}}{\partial \mathbf{v}} = \mathbf{A} \in \mathbb{R}^{m\times n} \tag{15}$$

Q: Solve

1. $\sum_j \delta_{ik} A_{ij=}$
2. $\sum_i \sum_j A_{ij}\delta_{lm}B_{il} =$
3. $\frac{\partial f}{\partial x_2}$, for $f_j(x_j) = \alpha x_j$ and $j_{1,2\ldots 10}$

4.4 Product rule & chain rule for vectors

The vector equivalent of the product rule is not terribly interesting, it generalises just how one might expect it to. However, taking componentwise derivatives and applying the chain rule carries a subtlety: if the intermediary function is also vector or matrix valued, we need to sum over it's components. Concretely:

$$\frac{\partial f_i}{\partial v_j} = \sum_k \frac{\partial f_i}{\partial g_k}\frac{\partial g_k}{\partial v_j} \tag{16}$$

This makes sense, since we need to consider all 'paths' through $\mathbf{g}$ through which input v_j might interact with output f_i.

4.4.1 A bit of backpropagation

Neural networks are trained using backpropagation. This amounts to little more than cleverly applying the chain rule to get gradients of the loss function with respect to the parameters. Let's take a simple neural network layer and find the gradient of its weight matrix. Since the network output is a vector and we are taking the gradient with respect to a matrix, this derivation will yield a three-dimensional object! This is a little different from the example in the lectures, but we did not have an example with a matrix derivative yet, so we thought we'd adapt it a little bit. Let's get started, shall we?

We are looking for $\frac{\partial}{\partial \mathbf{W}}\text{ReLU}(\mathbf{Wx} + \mathbf{b})$. The ReLU function is a common activation function, with $\text{ReLU}(x) = max(0, x)$. Its derivative whenever $x \leq 0$

is zero[2], and its derivative whenever $x > 0$ is one. We can model this derivative by the indicator function, which is:

$$\mathbb{I}_{x>0} = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{else} \end{cases} \tag{17}$$

So with that, let's get started and apply the chain rule:

$$\frac{\partial}{\partial W_{jk}} \text{ReLU}(\mathbf{W}\mathbf{x} + \mathbf{b})_i = \sum_l \frac{\partial \text{ReLU}(u)_i}{\partial u_l} \frac{\partial u_l}{\partial W_{jk}} \tag{18}$$

The first term yields:

$$\frac{\partial \text{ReLU}(u)_i}{\partial u_l} = \mathbb{I}_{u_i>0} \delta_{il} \tag{19}$$

The second term yields:

$$\begin{aligned} \frac{\partial u_l}{\partial W_{jk}} &= \frac{\partial}{\partial W_{jk}} \sum_m W_{lm} x_m + b_l & (20) \\ &= \sum_m x_m \delta_{jl} \delta_{km} & (21) \\ &= x_k \delta_{jl} & (22) \end{aligned}$$

Note the double Kronecker delta due to the two indices of the weight matrix. Combining these results, we get:

$$\begin{aligned} \frac{\partial}{\partial W_{jk}} \text{ReLU}(\mathbf{W}\mathbf{x} + \mathbf{b})_i &= \sum_l \mathbb{I}_{u_i>0} x_k \delta_{il} \delta_{jl} & (23) \\ &= \mathbb{I}_{u_i>0} x_k \delta_{ij} & (24) \end{aligned}$$

That was weird! The two Kronecker delta's combined to form a new one. This is because $\delta_{il}\delta_{jl}$ will only be non-zero for a given index l if $i = j$.

As you might have seen, our final answer contains three indices, making it a multi-dimensional array or matrix. Note that it is often called a tensor, which is not entirely correct, since it does not have an associated transformation rule.

[2]There exists mathematical justification for what happens at $x = 0$, which we will omit here.

Now, for inferring the shape, we have a bit of a notational quandary as we need to notate a three-dimensional object in two-dimensions. Using the conventions of the MML book (equation 5.87), we could write it as a vector with matrix-valued entries. Noting that the Kronecker delta is an identity matrix, we get:

$$\frac{\partial}{\partial \mathbf{W}} \text{ReLU}(\mathbf{W}\mathbf{x} + \mathbf{b}) = \mathbf{x} \circ \begin{bmatrix} \text{Diag}(\mathbb{I}_{\mathbf{W}\mathbf{x}+\mathbf{b}>0}) \\ \text{Diag}(\mathbb{I}_{\mathbf{W}\mathbf{x}+\mathbf{b}>0}) \\ \vdots \\ \text{Diag}(\mathbb{I}_{\mathbf{W}\mathbf{x}+\mathbf{b}>0}) \end{bmatrix} \tag{25}$$

$$= \begin{bmatrix} x_1 \text{Diag}(\mathbb{I}_{\mathbf{W}\mathbf{x}+\mathbf{b}>0}) \\ x_2 \text{Diag}(\mathbb{I}_{\mathbf{W}\mathbf{x}+\mathbf{b}>0}) \\ \vdots \\ x_K \text{Diag}(\mathbb{I}_{\mathbf{W}\mathbf{x}+\mathbf{b}>0}) \end{bmatrix} \tag{26}$$

Here Diag is a function that puts a vector on the diagonal of a square matrix and $\circ$ is the Hadamard or elementwise product. This means that we multiply every element of the vector $\mathbf{x}$ with a matrix and stack these in a column vector. This is strange notation, which is a consequence of high-dimensional nature of our object. Index notation can always be used since, together with our shape conventions, uniquely describe the final answer. Note, though, that when implementing this, you can simplify the computations by exploiting the diagonal nature of the matrix.

This was a tough derivation for this course. The exercises will generally be a bit easier than this one.

5 A relevant example: linear regression

At this point, the world's your oyster. You have all the tricks you need to compute any vector derivative, be they of—or with respect to—matrices, vectors or even stranger objects. Let's do some linear regression to put our skills into practice, shall we?

Assume we have N data points $\mathbf{x}_i \in \mathbf{R}^d$ and we stack them in a big matrix $X \in \mathbb{R}^{N \times d}$. The regression target $y_i \in \mathbb{R}$ can be similarly stacked to form $Y \in \mathbb{R}^N$. The trainable parameters $\beta \in \mathbb{R}^d$ are a set of numbers that combine inputs linearly and aim to map it close to the target value. We want our predictions $X\beta$ to be as close as possible to our targets Y. This implies that our ideal coefficients can be found by solving

$$\beta^* = \arg\min_{\beta} ||X\beta - Y||^2 \tag{27}$$

So let's do that! Let's take the derivative and simplify the norm:

$$\frac{\partial}{\partial\beta}||X\beta - Y||^2 = \frac{\partial}{\partial\beta}\left(\mathbf{X}\beta - \mathbf{Y}\right)^T\left(\mathbf{X}\beta - \mathbf{Y}\right) \tag{28}$$

$$= \frac{\partial}{\partial\beta}\left[\beta^T\mathbf{X}^T\mathbf{X}\beta - \beta^T\mathbf{X}^T Y - Y^T\mathbf{X}\beta + \mathbf{Y}^T\mathbf{Y}\right] \tag{29}$$

Pfew, this is quite the expression. In order to keep things simple, I'll split it up and we'll fill in the separate parts as soon as we have them. First, let's take a look at the term where β appears twice:

$$\frac{\partial}{\partial x_i}\sum_{j,k} x_j A_{jk} x_k = \sum_{j,k} A_{jk} x_k \delta_{ij} + A_{jk} x_j \delta_{ik} \tag{30}$$

$$= \sum_j A_{ji} x_j + \sum_k A_{ik} x_k \tag{31}$$

This implies that

$$\frac{\partial}{\partial\mathbf{x}}\mathbf{x}^T\mathbf{A}\mathbf{x} = \mathbf{x}^T(\mathbf{A} + \mathbf{A}^T) \tag{32}$$

$$= 2\mathbf{x}^T\mathbf{A} \quad \text{if A is symmetric.} \tag{33}$$

If that went a little too fast, let me explain. First note that both sums in equation 30 imply that there is some sort of matrix-vector multiplication going on. Given our shape convention, the output should be a row vector, so they could either be $\mathbf{x}^T\mathbf{A}$ or $\mathbf{x}^T\mathbf{A}^T$. Looking at the indices that are summed over in that same equation, we see that the left hand term sums over the first index, whereas the right hand term sums over the second. This implies that the former is $\mathbf{A}$ and the latter is $\mathbf{A}^T$. It does not really matter which is which since we sum them, but this is how I would identify them based on componentwise notation.

Next, let's look at the term where β appears once:

$$\frac{\partial}{\partial x_i}\sum_{j,k}\mu_k A_{jk} x_k = \sum_{j,k} A_{jk}\mu_j \frac{\partial x_k}{\partial x_i} \tag{34}$$

$$= \sum_{j,k} A_{jk}\mu_j \delta_{ki} \tag{35}$$

$$= \sum_j A_{ji}\mu_j, \tag{36}$$

which implies that

$$\frac{\partial}{\partial x}\mu^T\mathbf{A}\mathbf{x} = \mu^T\mathbf{A} \tag{37}$$

Note that any identity of the form vector^T-matrix-vector is a scalar. This means that we are free to take its transpose, since the transpose of a scalar is a scalar. This means that $\mathbf{Y}^T\mathbf{X}\beta = \beta^T\mathbf{X}^T\mathbf{Y}$. Using this information, our derivative can be found as:

$$\frac{\partial}{\partial\beta}\left[\beta^T\mathbf{X}^T\mathbf{X}\beta - \beta^T\mathbf{X}^TY - Y^T\mathbf{X}\beta + \mathbf{Y}^T\mathbf{Y}\right] = 2\mathbf{X}^T\mathbf{X}\beta - 2\mathbf{X}^T\mathbf{Y} \tag{38}$$

$$= 0 \tag{39}$$

After taking the transpose and dividing by two, we see that the ideal parameters for our linear regression are found by

$$\mathbf{X}^T\mathbf{X}\beta^* = \mathbf{X}^T\mathbf{Y} \tag{40}$$

$$\beta^* = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y}. \tag{41}$$

Where we multiplied from the left with the inverse to solve for β^*.

6 Final remarks

We hope that you have found this document useful and that it will help you during this course. You now possess the tools to derive all the identities in the MML-book concerning derivatives, which is kinda neat. For homework assignments, you are free to use component notation or to refer to equations from the MML book or the matrix cookbook.

If you spot any errors in this document, please message Rob at r.d.hesselink@uva.nl, so we may improve it.