Math Notation

In deep learning we encounter many different fields and each has its own notation. We will stick to the following conventions throughout most of this course, and note it when we deviate from these rules. To define the conventions we give examples of usage, from which you can infer the pattern.

General notation

- Scalar: x, y, z.
- Vector: $\mathbf{x}, \mathbf{y}, \mathbf{z}$. We use bold letters to represent vectors, matrices, and tensors.
- Index of a vector: x_i, x_j, y_i, or $x[i]$, $x[j]$, $y[i]$.
- Matrix: $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$. We use bold letters to represent vectors, matrices, and tensors.
- Index of a matrix: X_{ij}, Y_{jk}, Z_{ii}, or $X[i, j]$, $Y[j, k]$, $Z[i, i]$.
- For an indexed matrix X_{ij} or $X[i, j]$, i indexes rows and j indexes columns. We use non bold font because X_{ij} and $X[i, j]$ are scalars.
- Slice of a matrix: $\mathbf{X}_i$ or $\mathbf{X}[i, :]$; $\mathbf{X}[:, j]$. Here is one example:

$$\mathbf{X} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \qquad \mathbf{X}[2, :] = \begin{bmatrix} 3 & 4 \end{bmatrix}$$

- Tensor (i.e., multidimensinoal arrays): Typically, we will use lowercase bold variables to represent tensors, for example, $\mathbf{x}$. This is because tensors can have any number of dimensions (they can be one-dimensional, two-dimensional, three-dimensional, and so on). Furthermore, we will often define operators that are agnostic to the dimensionality of the tensor (they apply to N-dimensional arrays, for any N). However, in some sections, we use uppercase to make a distinction between tensors of different shapes, and we will specify when this is the case.
- Index or slice of a tensor: $x[c, i, j, k]$, $\mathbf{x}[:, :, k]$
- A set of N datapoints: $\{x^{(i)}\}_{i=1}^N$, $\{\mathbf{x}^{(i)}\}_{i=1}^N$, $\{\mathbf{X}^{(i)}\}_{i=1}^N$
- Dot product: $\mathbf{x}^\mathsf{T}\mathbf{y}$
- Matrix product: $\mathbf{AB}$
- Hadamard product (i.e., element-wise product): $\mathbf{x} \odot \mathbf{y}$, $\mathbf{A} \odot \mathbf{B}$
- Product of two scalars: ab or $a * b$

Math Notation

Machine learning

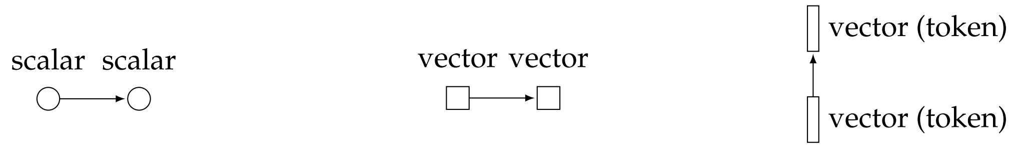
- Loss function: $\mathcal{L}$ (usually this is a per-datapoint loss)
- Total cost over all datapoints: J
- Generic learnable parameters: θ

Neural nets

- *Parameters*: θ; these include weights, $\mathbf{W}$, and biases, $\mathbf{b}$, as well as any other learnable parameters of a network.
- *Data*: $\mathbf{x}$ – "Data" can refer to inputs to the network, activations on hidden layers, outputs from the network, etc. Any representation of the signal being processed is considered to be "data". Sometimes we will wish to distinguish between the raw inputs, hidden units, and outputs to a network, in which case we will use $\mathbf{x}$, $\mathbf{h}$, and $\mathbf{y}$ respectively. When we need to to distinguish between pre-activation hidden units and post-activation, we will use $\mathbf{z}$ for pre-activation and $\mathbf{h}$ for post-activation.
- When describing batches of tensors (as is commonly encountered in code), you may encounter $x_l[b, c, n, m]$ to represent an activation on layer l of some network, with b indexing batch element, n and m as spatial coordinates, and c indexing channels.
- Neuron values on layer l of a deep net: $\mathbf{x}_l$
- $x_l[n]$ refers to the n-th neuron on layer l. For neural networks with spatial feature maps (such as convolutional neural networks), each layer is an array of neurons, and we will use notation such as $x_l[n, m]$ to index the neuron at location n, m.
- The layer l neural representation of the i-th datapoint in a dataset is written as $\mathbf{x}_l^{(i)}$.
- We will also use $\mathbf{x}_{\texttt{in}}$ and $\mathbf{x}_{\texttt{out}}$ when we are describing a particular layer or module in a neural net and wish to speak of its inputs and outputs without having to keep track of layer indices.
- For signals with multiple channels, including neural network feature maps, the first dimension of the tensor indexes over channel. For example, in $\mathbf{x} \in \mathbb{R}^{C \times N \times M \times \dots}$, where C is the number of channels of the signal.
- For transformers, we deviate from the previous point slightly, in order to match standard notation: a set of tokens (which will be defined in the transformers chapter) is represented by a $[N \times d]$ matrix, where d is the token dimensionality.

Math Notation

We will use circles to represents neurons, which are scalar nodes in a graph. Edges between neurons represent a $\mathbb{R} \rightarrow \mathbb{R}$ transformation (usually parameterized by a single weight associated with the edge). In many neural net architectures, the nodes of our networks are not individual neurons, but instead may be multidimensional vectors of neurons, which we will draw as squares or rectangles. In these cases, an edge between a node of dimensionality C_1 and a node of dimensionality C_2 represents a $\mathbb{R}^{C_1} \rightarrow \mathbb{R}^{C_2}$ transformation (which may be parameterized by multiple weights). The three node symbols we use are shown below:



A token is a vector of neurons used in a particular way, which will be defined in the transformers chapter. We give it a special symbol, a rectangle, to distinguish it from other kinds of vectors of neurons.

As shown above, sometimes we draw networks with the layers moving left to right and sometimes bottom to top. Both mean the same thing, and the direction in each figure is just chosen for visual clarity.

Probabilities
We will typically not distinguish between random variables and realizations of those variables; which we mean should be clear from context. When it is important to make a distinction, we will use non-bold capital letters to refer to random variables and lowercase to refer to realizations.

Suppose X, Y are discrete random variables and $\mathbf{x}, \mathbf{y}$ are realizations of those variables. X and Y may take on values in the sets $\mathcal{X}$ and $\mathcal{Y}$ respectively.

- $a = p(X = \mathbf{x} | \dots)$ is the probability of the realization $X = \mathbf{x}$, possibly conditioned on some observations (a is a scalar).
- $f = p(X | \dots)$ is the probability distribution over X, possibly conditioned on some observations (f is a function: $f : \mathcal{X} \rightarrow \mathbb{R}$). If $\mathcal{X}$ is discrete, f is the *probability mass function*. If $\mathcal{X}$ is continuous, f is the *probability density function*.
- $p(\mathbf{x} | \dots)$ is shorthand for $p(X = \mathbf{x} | \dots)$.
- and so forth, following these patterns.
- Suppose we have defined a named distribution, e.g., p_θ; then referring to p_θ on its own is shorthand for $p_\theta(X)$

For continuous random variables, all the above notations hold except that they refer to probability densities and probability density functions rather than probabilities and probability distributions. We will sometimes use the term "probability distribution" when referring to continuous distributions, and in those cases this should be understood to refer to a probability density function.

Matrix calculus conventions In these notes, we adopt the following conventions for matrix calculus. These conventions make the equations simpler, and that also means simpler implementations when it comes to actually writing these equations in code. Everything in this section is just definitions. There is no right or wrong to it. We could have used other conventions but we will see that these are useful ones.

Vectors are represented as column vectors with shape $[N \times 1]$:

$$\mathbf{x} \triangleq \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \tag{1}$$

If y is a scalar and $\mathbf{x}$ is an N-dimensional vector, then the gradient $\frac{\partial y}{\partial \mathbf{x}}$ is a row vector of shape $[1 \times N]$:

$$\frac{\partial y}{\partial \mathbf{x}} \triangleq \begin{bmatrix} \frac{\partial y}{\partial x_1} & \frac{\partial y}{\partial x_2} & \cdots & \frac{\partial y}{\partial x_N} \end{bmatrix} \tag{2}$$

If $\mathbf{y}$ is an M-dimensional vector and $\mathbf{x}$ is a N-dimensional vector then the gradient (also called the Jacobian in this case) is shaped as $[M \times N]$:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \triangleq \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_N} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial y_M}{\partial x_1} & \frac{\partial y_M}{\partial x_2} & \cdots & \frac{\partial y_M}{\partial x_N} \end{bmatrix} \tag{3}$$

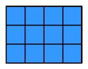
Finally, if $\mathbf{W}$ is an $[N \times M]$ dimensional matrix, and $\mathcal{L}$ is a scalar, then the gradient $\frac{\partial \mathcal{L}}{\mathbf{W}}$ is represented as an $[M \times N]$ dimensional matrix (note that the dimensions are transposed from what you might have expected; this makes the math simpler later):

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} \triangleq \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \mathbf{W}_{11}} & \cdots & \frac{\partial \mathcal{L}}{\partial \mathbf{W}_{N1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{L}}{\partial \mathbf{W}_{1M}} & \cdots & \frac{\partial \mathcal{L}}{\partial \mathbf{W}_{NM}} \end{bmatrix} \tag{4}$$

Math Notation

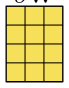
We will sometimes draw matrices and vectors to help visualize the operations. For example, if $N = 3$ and $M = 4$:

$\mathbf{W}$



then, the gradient will have the form:

$\frac{\partial J}{\partial \mathbf{W}}$



Conventions that will not be strictly adhered to

- We will often use $\mathbf{x}$ as the input to a function, and $\mathbf{y}$ as the output.
- f, g, and h are typically functions. The corresponding function spaces are $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$.

Miscellaneous

- The word **dimension** has two usages in the computational sciences. The first usage is as a coordinate in a multivariate data structure, for example, "the i-th dimension of a vector" or "a 128-dimensional feature space." The second usage is as the shape of a multidimensional array, as in "a 4D tensor." We will use both these meanings in this book and we hope the usage will be clear from context.

MIT OpenCourseWare
https://ocw.mit.edu

6.7960 Deep Learning
Fall 2024

For information about citing these materials or our Terms of Use, visit: https://ocw.mit.edu/terms