Derivatives, Gradients, Jacobians

From basic calculus we are (hopefully) familiar with the *derivative*

$$\frac{\partial y}{\partial x}$$

where $y = f(x)$ for some univariate functions f.

But what about

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$$

where $\mathbf{y} = f(\mathbf{x})$ is a multivariate function (on vector $\mathbf{x}$) with range that is *also* a vector.

In general, $\mathbf{y}$ and $\mathbf{x}$ may be vectors and we need to define the *Jacobian* $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$

Before giving the general form for the Jacobian, we illustrate it in steps

Scalar y, vector $\mathbf{x}$

$$\frac{\partial y}{\partial \mathbf{x}}$$

- is the vector of length $|\mathbf{x}|$ of defined as

$$\left(\frac{\partial y}{\partial \mathbf{x}}\right)_j = \frac{\partial y}{\partial \mathbf{x}_j}$$

Example

$|\mathbf{x}| = 2$ and $y = \mathbf{x}_1 * \mathbf{x}_2$

$$\begin{aligned} \frac{\partial y}{\partial \mathbf{x}} &= \begin{pmatrix} \frac{\partial y}{\partial \mathbf{x}_1} & \frac{\partial y}{\partial \mathbf{x}_2} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{x}_2 & \mathbf{x}_1 \end{pmatrix} \end{aligned}$$

To be even more concrete: consider a Regression Task using the Mean Squared Error (MSE) loss function.

$$\mathcal{L}_\Theta = \text{MSE}(\mathbf{y}, \hat{\mathbf{y}}, \Theta) = \frac{1}{m} \sum_{i=1}^{m} (\mathbf{y}^{(\mathbf{i})} - \hat{\mathbf{y}}^{(\mathbf{i})})^2$$

Using Θ to denote the vector of parameters

- Θ_0 is the intercept
- Θ_j is the sensitivity of the loss to the independent variable (feature) j

The derivative (gradient) of the scalar $\mathcal{L}_\Theta$ with respect to vector Θ is:

$$\nabla_\Theta \mathcal{L}_\Theta = \begin{pmatrix} \frac{\partial}{\partial \Theta_0} \mathrm{MSE}(\mathbf{y}, \hat{\mathbf{y}}, \boldsymbol{\Theta}) \\ \frac{\partial}{\partial \Theta_1} \mathrm{MSE}(\mathbf{y}, \hat{\mathbf{y}}, \boldsymbol{\Theta}) \\ \vdots \\ \frac{\partial}{\partial \Theta_n} \mathrm{MSE}(\mathbf{y}, \hat{\mathbf{y}}, \boldsymbol{\Theta}) \end{pmatrix}$$

Here are the details of the derivative of $\mathcal{L}_\Theta$ with respect to independent variable j

$$
\begin{array}{lcll}
\frac{\partial}{\partial \Theta_j}\mathrm{MSE}(\mathbf{y}, \hat{\mathbf{y}}, \boldsymbol{\Theta}) & = & \frac{1}{m}\sum_{i=1}^m \frac{\partial}{\partial \Theta_j}(\mathbf{y}^{(\mathbf{i})} - \hat{\mathbf{y}}^{(\mathbf{i})})^2 & \text{definition} \\
 & = & \frac{1}{m}\sum_{i=1}^m 2 * (\mathbf{y}^{(\mathbf{i})} - \hat{\mathbf{y}}^{(\mathbf{i})}) \frac{\partial}{\partial \Theta_j} \hat{\mathbf{y}}^{(\mathbf{i})} & \text{chain rule} \\
 & = & \frac{1}{m}\sum_{i=1}^m 2 * (\mathbf{y}^{(\mathbf{i})} - \hat{\mathbf{y}}^{(\mathbf{i})}) \frac{\partial}{\partial \Theta_j} (\boldsymbol{\Theta} * \mathbf{x}^{(\mathbf{i})}) & \hat{\mathbf{y}}^{(\mathbf{i})} = \boldsymbol{\Theta}^T \cdot \mathbf{x}^{(\mathbf{i})} \\
 & = & \frac{1}{m}\sum_{i=1}^m 2 * (\mathbf{y}^{(\mathbf{i})} - \hat{\mathbf{y}}^{(\mathbf{i})}) \mathbf{x}_j^{(\mathbf{i})} & \\
 & = & \frac{2}{m}\sum_{i=1}^m (\mathbf{y}^{(\mathbf{i})} - \hat{\mathbf{y}}^{(\mathbf{i})}) \mathbf{x}_j^{(\mathbf{i})} &
\end{array}
$$

Vector $\mathbf{y}$, scalar $\boldsymbol{x}$

$$\frac{\partial \mathbf{y}}{\partial x}$$

- is a column vector with $|\mathbf{y}|$ rows
- defined as

$$\left(\frac{\partial \mathbf{y}}{\partial x} \right)^{(\mathbf{i})} = \frac{\partial \mathbf{y}^{(\mathbf{i})}}{\partial x}$$

Technically (and this will be important when we define higher dimensional gradients recursively)

- is the vector of length 1
- whose *element* is a vector of length $|\mathbf{y}|$

Example $\mathbf{y} = (x^0, x^1, x^2)$

$$\begin{aligned}
\frac{\partial \mathbf{y}}{\partial x} &= \begin{pmatrix} \frac{\partial \mathbf{y}^{(1)}}{\partial x} \\ \frac{\partial \mathbf{y}^{(2)}}{\partial x} \\ \frac{\partial \mathbf{y}^{(3)}}{\partial x} \end{pmatrix} \\
&= \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix}
\end{aligned}$$

Vector $\mathbf{y}$, vector $\mathbf{x}$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$$

- is the vector of length $|\mathbf{x}|$
- whose *element* is a vector of length $|\mathbf{y}]$
- defined as

$$\left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)_j^{(\mathbf{i})} = \frac{\partial \mathbf{y}^{(\mathbf{i})}}{\partial \mathbf{x}_j}$$

Example $|\mathbf{x}| = 2, y = (\mathbf{x}_1 + \mathbf{x}_2, \mathbf{x}_1 * \mathbf{x}_2)$

$$\begin{aligned}\frac{\partial \mathbf{y}}{\partial \mathbf{x}} &= \begin{pmatrix} \frac{\partial \mathbf{y}^{(1)}}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{y}^{(1)}}{\partial \mathbf{x}_2} \\ \frac{\partial \mathbf{y}^{(2)}}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{y}^{(2)}}{\partial \mathbf{x}_2} \end{pmatrix} \\ &= \begin{pmatrix} 1 & 1 \\ \mathbf{x}_2 & \mathbf{x}_1 \end{pmatrix}\end{aligned}$$

Tensors and Generalized Jacobians

A *tensor* is multi-dimensional collection of values.

We are familiar with special cases

- a vector is a tensor with 1 dimension
- a matrix is a tensor with 2 dimensions

A D-dimensional tensor is a collection of numbers with *shape*

$$(n_1 \times n_2 \times \ldots \times n_D)$$

We can define the *Generalized Jacobian*

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$$

analogous to how we defined the Jacobian.

The main difference is that now the indices i and j change from *scalars* to *tensors*

Let

- the shape of $\mathbf{x}$ be $(n_{x_1} \times n_{x_2} \times \dots n_{x_{D_x}})$
- the shape of $\mathbf{y}$ be $(n_{y_1} \times n_{y_2} \times \dots n_{y_{D_y}})$

$$\left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)^{\mathbf{(i)}}_{j}$$

Note that

- the number of dimensions of $\mathbf{y}^{(\mathbf{i})}$ is $|\mathbf{y}| - 1$
- the number of dimensions of $\mathbf{x}_j$ is $|\mathbf{x}| - 1$

so the recursive call (RHS of equation) operates on an object of lesser dimension and hence will reduce to a base case (derivatives involving only vectors and scalars)

Where do these higher dimensional tensors come from ?

They are omnipresent !

- The mini batch index
- multi-dimensional input data

Mini batch index

When TensorFlow shows you the shape of an object, it typically has one more dimension than "natural" and the leading dimension is `None`.

That is because TensorFlow computes *on every element of the mini batch* simultaneously.

So the leading index points to an input example.

Hence the extra dimension.

Multidimensional data

Lots of data is multi-dimensional.

For examples images have a height, width and depth (number of color channels).

Before we introduced Tensors, we "flattened" higher dimensional images into vectors.

We then had to "unflatten" the scalar derivatives in order to rearrange them so as to correspond to the same index in the input from which they originated.

For the most part, this flatten/unflatten paradigm is not necessary if we operate over Tensors.

Conclusion

The derivatives that are needed for Gradient Descent often involve tensors.

This module formalized what it means to take derivatives of higher dimensional objects.

In [4]:

```python
print("Done")
```

```
Done
```