Multivariable Calculus Review

We will be covering some of the most relevant concepts from multivariable calculus for this course, and show how they can be extended to deal with matrices of training data.

Partial Derivatives

The derivative of a function of 2 variables $f(x, y)$ w.r.t. either one of its variables is defined as

$$\frac{\partial}{\partial x} f(x, y) := \lim_{h \to 0} \frac{f(x + h, y) - f(x, y)}{h}$$
$$\frac{\partial}{\partial y} f(x, y) := \lim_{h \to 0} \frac{f(x, y + h) - f(x, y)}{h}$$

This is a very obvious extension going from the derivative definition for functions of single variables.

As an example, what is $\frac{\partial}{\partial x} f(x, y)$ where $f(x, y) = x^2 cos(xy)$?

$$\frac{\partial}{\partial x} f(x, y) = 2x cos(xy) - x^2 sin(xy) y$$

Directional Derivatives

Before we define the gradient, we discuss the closely related concept of a directional derivative.

The directional derivative of a function $f(x, y)$ at (x_0, y_0) moving in the direction of a unit vector $u = [a, b]$ is

$$D_u f(x_0, y_0) := \lim_{h \to 0} \frac{f(x_0 + ha, y_0 + hb) - f(x_0, y_0)}{h}$$

This simply tells us how the scalar output of f will change when we move in an arbitrary direction that is not necesarily axis-aligned. It can be written as

$$D_u f(x, y) = \left[\frac{\partial}{\partial x} f(x, y), \frac{\partial}{\partial y} f(x, y) \right] \cdot u$$

Gradient

This is perhaps the term you will hear most often in the context of neural networks. The gradient is informally defined as the vector pointing in the direction of steepest increase for a given function. Thus, if we would like to minimize a loss function such as mean squared error (MSE), we move in the opposite direction of the gradient, i.e. in the direction of the negative gradient.

Now the gradient is simply defined as

$$\nabla f(x, y) := \left[\frac{\partial}{\partial x} f(x, y), \frac{\partial}{\partial y} f(x, y)\right]$$

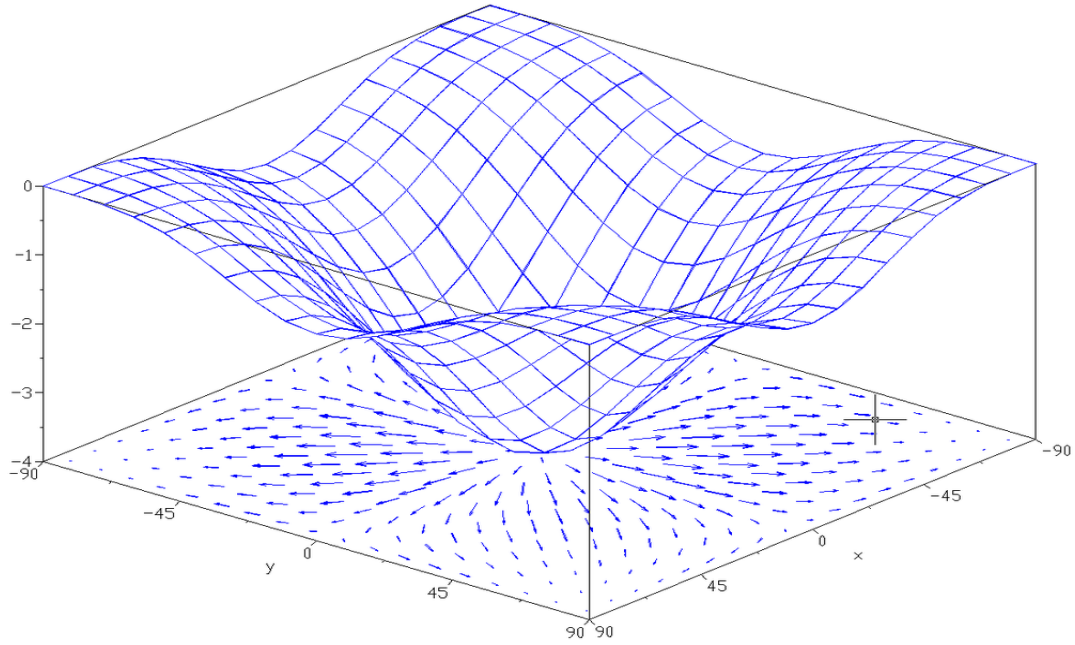
Note that this is a vector-field (vector-valued function), and not a scalar-valued function!

What does $\nabla f(x, y)$ have to do with neural networks and minimizing loss functions? Let's see in which direction the directional derivative achieves the highest possible value. Using the gradient notation, we can rewrite the directional derivative as

$$\begin{aligned} D_u f(x, y) &= \nabla f(x, y) \cdot u \\ &= ||\nabla f(x, y)||_2 \cdot ||u||_2 \cdot cos(\theta) \\ &= ||\nabla f(x, y)||_2 \cdot cos(\theta) \end{aligned}$$

This quantity is maximized when $cos(\theta) = 1$, i.e. $\nabla f(x, y)$ and u are parallel. Therefore, the gradient points in the direction of steepest increase for f.

So if f is a neural network parameterized by some weights w, and $x \in \mathbb{R}^D$ is some input to it, then taking steps in $\nabla_w f_w(x)$ will move us in the direction that maximizes the output of f. Thus, if f is a binary classification network, say the probability of an image being of a car, then moving in this direction would maximimally increase the probability of x being classified as a car.



Basic ML Setup

If you recall from CSC411, the basic supervised learning scenarios are regression and classification. Most buzz in computer vision has driven by success on classifcation problems, so we start with that first.

Classification

In K-class classification, we are given a dataset $\{\mathbf{x}^{(i)}, t^{(i)}\}_{i=1}^{N}$ of N samples where $\mathbf{x}^{(i)} \in \mathbb{R}^D$ and $t^{(i)} \in [1, \ldots, K]$.

We would like to fit a probabilistic model, in our case a neural network, that maximizes the probability

$$\prod_{i=1}^{N} p(t^{(i)}|\mathbf{x}^{(i)}, \mathbf{w})$$

we can vectorize these objects/terms by defining

$$\mathbf{X} = \begin{bmatrix} (\mathbf{x}^{(1)})^\top \\ \vdots \\ (\mathbf{x}^{(N)})^\top \end{bmatrix}$$

$$\mathbf{T} = \begin{bmatrix} (t^{(1)})^\top \\ \vdots \\ (t^{(N)})^\top \end{bmatrix}$$

where $\mathbf{X}$ is called the design matrix, and has dimensions (N, D), where entry $\mathbf{X}_{ij}$ corresponds to the j-th feature for the i-th training sample. $\mathbf{T}$ is our matrix of labels. The i-th row of $\mathbf{T}$ is the one-hot encoded vector of the label for the i-th sample which is simply a K-dimensional vector of all 0s except for the a single 1 in the position of the label. If $K = 4$, and $t^{(i)} = 2$, the one-hot encoded version is

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

We assume that all vectors are column vectors.

Our goal becomes to maximize the probability

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}) = \prod_{i=1}^{N} p(t^{(i)}|\mathbf{x}^{(i)}, \mathbf{w})$$
$$= \prod_{i=1}^{N}\prod_{k=1}^{K} p(k|\mathbf{x}^{(i)}, \mathbf{w})^{\mathbf{T}_{ik}}$$

Note that just the term in the innermost product corresponding to the true label of the i-th can be less than 1. Since this expression would lead to numerical underflow given a large enough dataset, we can equivalently maximize a monotonically increasing function of this expression, i.e. take the log

$$\log\left(\prod_{i=1}^{N}\prod_{k=1}^{K} p(k|\mathbf{x}^{(i)}, \mathbf{w})^{\mathbf{T}_{ik}}\right) = \sum_{i=1}^{N}\sum_{k=1}^{K} \mathbf{T}_{ik}\log(p(k|\mathbf{x}^{(i)}, \mathbf{w}))$$

If we negate this term, we end up with the most popular loss function for classification known as cross-entropy

$$L_{CE} = -\sum_{i=1}^{N}\sum_{k=1}^{K} \mathbf{T}_{ik}\log(p(k|\mathbf{x}^{(i)}, \mathbf{w}))$$

Our goal is to **minimize** L_{CE} by moving in the direction $-\nabla_{\mathbf{w}} L_{CE}$.

Linear Regression

Multivariate input. Univariate output. $\mathbb{R}^M \rightarrow \mathbb{R}$

Forward pass

Suppose we have some features $x_1, x_2, \ldots, x_M$, and we predict a single scalar $y = w_1 x_1 + w_2 x_2 + \cdots + w_M x_M$. We can write this in matrix notation as a dot product

$$y = \sum_{i=1}^{M} w_i x_i = \mathbf{w}^T\mathbf{x} \quad \text{where} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix}$$

We say $\mathbf{x} \in \mathbb{R}^M$ to indicate that $\mathbf{x}$ is a M-dimensional vector.

Backward pass

Suppose we have a target t and a loss function $L = (y - t)^2$, what is the gradient $\nabla_{\mathbf{w}} L$?

Remember the definition of the gradient,

$$\nabla_w L = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \vdots \\ \frac{\partial L}{\partial w_M} \end{bmatrix}$$

Each element is a partial derivative that can be written out using *chain rule*.

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w_i} = 2(y - t)\ x_i$$

So the gradient is

$$\nabla_w L = \begin{bmatrix} 2(y - t)\ x_1 \\ \vdots \\ 2(y - t)\ x_M \end{bmatrix} = 2(y - t)\mathbf{x}$$

Remark. There are some matrix calculus formulas that are worth remembering. One that we could have used here is

$$\mathbf{w}^T \mathbf{x} \implies \nabla_{\mathbf{w}} \mathbf{w}^T \mathbf{x} = \mathbf{x}$$

So the math becomes much easier.

$$\nabla_{\mathbf{w}} L = \underbrace{(\nabla_y L)(\nabla_{\mathbf{w}} y)}_{\text{Chain rule holds for matrix calculus.}} = 2(y - t)\mathbf{x}$$

Multivariate input. **Multivariate output**. $\mathbb{R}^D \rightarrow \mathbb{R}^M$

Forward pass

Suppose we have some features $x_1, x_2, \ldots, x_D$, and we predict a **vector** $\mathbf{y} \in \mathbb{R}^{\mathbf{M}}$ where $y_j = w_{j1}x_1 + w_{j2}x_2 + \cdots + w_{jD}x_D$. We can write this in matrix notation as a **matrix-vector product**.

$$\mathbf{y} = \begin{bmatrix} w_{11}x_1 + w_{12}x_2 + \cdots + w_{1D}x_D \\ w_{21}x_1 + w_{22}x_2 + \cdots + w_{2D}x_D \\ \vdots \\ w_{M1}x_1 + w_{M2}x_2 + \cdots + w_{MD}x_D \end{bmatrix} = \begin{bmatrix} w_1^T x \\ w_2^T x \\ \vdots \\ w_M^T x \end{bmatrix} = \mathbf{W}\mathbf{x},$$

Where

$$\mathbf{W} = \begin{bmatrix} w_1^T \\ w_2^T \\ \vdots \\ w_M^T \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \ldots & w_{1D} \\ w_{21} & w_{22} & \ldots & w_{2D} \\ \vdots & & & \\ w_{M1} & w_{M2} & \ldots & w_{MD} \end{bmatrix}$$

We say $\mathbf{W} \in \mathbb{R}^{M \times D}$ to indicate that $\mathbf{W}$ is a matrix of dimension M by D.

Backward pass

Suppose we have a target t and a loss function $L = ||y - t||_2^2$, what is the gradient $\nabla_{\mathbf{W}} L$?

Remember the definition of the gradient,

$$\nabla_w L = \begin{bmatrix} \frac{\partial L}{\partial w_{11}} & \frac{\partial L}{\partial w_{12}} & \cdots & \frac{\partial L}{\partial w_{1D}} \\ \frac{\partial L}{\partial w_{21}} & \frac{\partial L}{\partial w_{22}} & \cdots & \frac{\partial L}{\partial w_{2D}} \\ \vdots & & & \\ \frac{\partial L}{\partial w_{M1}} & \frac{\partial L}{\partial w_{M2}} & \cdots & \frac{\partial L}{\partial w_{MD}} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial \mathbf{w_1}}^T \\ \frac{\partial L}{\partial \mathbf{w_2}}^T \\ \vdots \\ \frac{\partial L}{\partial \mathbf{w_M}}^T \end{bmatrix}$$

In this particular case, *because* L *depends on* $\mathbf{w_1}$ *only through* y_1, we have

$$\frac{\partial L}{\partial \mathbf{w_k}} = \sum_{j=1}^{M} \frac{\partial L}{\partial y_j} \frac{\partial y_j}{\partial \mathbf{w_k}} = \frac{\partial L}{\partial y_k} \frac{\partial y_k}{\partial \mathbf{w_k}}$$

(ie. Can show that $\frac{\partial y_j}{\partial \mathbf{w_k}} = 0$ if $j \neq k$.)

The two parts:

$$\frac{\partial L}{\partial y_k} = \frac{\partial}{\partial y_k} \sum_{j=1}^{D} (y_j - t_j)^2 = 2(y_k - t_k)$$

$$\frac{\partial y_k}{\partial \mathbf{w_k}} = \frac{\partial}{\partial \mathbf{w_k}} \mathbf{w_k}^T \mathbf{x} = \mathbf{x}$$

And we finally have

$$\nabla_w L = \begin{bmatrix} \frac{\partial L}{\partial \mathbf{w_1}}^T \\ \frac{\partial L}{\partial \mathbf{w_2}}^T \\ \vdots \\ \frac{\partial L}{\partial \mathbf{w_M}}^T \end{bmatrix} = \begin{bmatrix} 2(y_1 - t_1)\mathbf{x}^T \\ 2(y_2 - t_2)\mathbf{x}^T \\ \vdots \\ 2(y_M - t_M)\mathbf{x}^T \end{bmatrix} = 2(\mathbf{y} - \mathbf{t})\mathbf{x}^T$$

Remark. There are some matrix calculus formulas that are worth remembering. One that we could have used here is

For any vectors $\mathbf{v} \in \mathbb{R}^M, \mathbf{A} \in \mathbb{R}^{M \times D}, \mathbf{u} \in \mathbb{R}^D$,

For any vector $\mathbf{v}$,

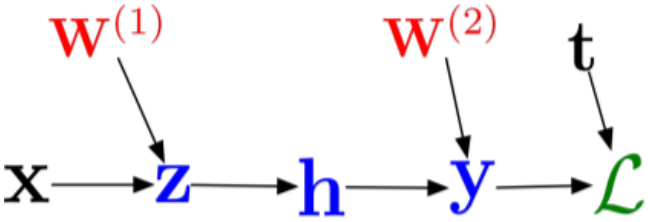
$$\nabla_{\mathbf{A}} \mathbf{v}^T \mathbf{A} \mathbf{u} = \mathbf{v}\mathbf{u}^T$$

So the math becomes much easier.

$$\nabla_{\mathbf{W}} L = \underbrace{(\nabla_{\mathbf{y}} L)^T (\nabla_{\mathbf{W}} \mathbf{y})}_{\text{Chain rule holds for matrix calculus.}} = \nabla_{\mathbf{W}} (\underbrace{2(\mathbf{y} - \mathbf{t})}_{\text{Evaluate } \nabla_{\mathbf{y}} L \text{ first.}})^T W \mathbf{x} = 2(\mathbf{y} - \mathbf{t})\mathbf{x}^T$$

1-Hidden Layer Neural Network

Putting it all together, suppose we now have a neural network with a single hidden layer.



No more summation notation. Let's put what we've learned about matrix notation in use.

$$(1)\quad \nabla_{\mathbf{w}} \mathbf{w}^T \mathbf{x} = \mathbf{x}$$
$$(2)\quad \nabla_{\mathbf{A}} \mathbf{v}^T \mathbf{A} \mathbf{u} = \mathbf{v}\mathbf{u}^T$$

Forward pass.

$$\mathbf{z} = \mathbf{W}^{(1)}\mathbf{x} \qquad \text{(Linear transformation)}$$
$$\mathbf{h} = \sigma(\mathbf{z}) \qquad \text{(Element-wise nonlinear function)}$$
$$\mathbf{y} = \mathbf{W}^{(2)}\mathbf{h} \qquad \text{(Linear transformation)}$$
$$L = ||\mathbf{y} - \mathbf{t}||^2 \qquad \text{(Loss function)}$$

Backward pass.

$$\nabla_{\mathbf{y}} L = 2(\mathbf{y} - \mathbf{t}) \qquad \text{(Pass gradient through loss function)}$$
$$\nabla_{\mathbf{h}} L = ((\nabla_{\mathbf{y}} L)^T \mathbf{W}^{(2)})^T \qquad \text{(Pass gradient through linear function; uses (1))}$$
$$\nabla_{\mathbf{W}^{(2)}} L = (\nabla_{\mathbf{y}} L)\mathbf{h}^T \qquad \text{(Pass gradient to } \mathbf{W}^{(2)} \text{ ; uses (2))}$$
$$\nabla_{\mathbf{z}} L = (\nabla_{\mathbf{h}} L) \circ \sigma'(\mathbf{z}) \qquad \text{(Pass gradient through nonlinearity)}$$
$$\nabla_{\mathbf{W}^{(1)}} L = (\nabla_{\mathbf{z}} L)\mathbf{x}^T \qquad \text{(Pass gradient to } \mathbf{W}^{(1)} \text{ ; uses (2))}$$