Algorithms for Multivariate Newton-Raphson for Optimization
When the Hessian Matrix is a Constant plus a Diagonal

Max Sklar
Local Maximum Labs
May 19, 2023

Abstract

This paper concerns the special case of multivariate Newton's Method for optimization where the Hessian Matrix is composed of a diagonal matrix plus a constant term. It derives an algorithm for a single Newton step given the diagonal, constant, and gradient. It also derives an alternative algorithm in *log-space* for functions acting on positive inputs. Additional steps and definitions are provided to guide those revisiting their knowledge in algorithms, linear algebra and vector calculus.

1 Introduction

Newton's method (or the Newton-Raphson method) is a way to approximate a solution to the equation $f(x) = 0$ for a differentiable function f. The method works by starting at an initial value x_0 and successfully updating it using the formula

$$x_{i+1} = x_i - \frac{f(x_0)}{f'(x_0)}.$$

This process continues until it converges, which is not guaranteed. When one wants to find a local maximum or local minimum of f, this is equivalent to finding the root of $f'(x)$. If $f'(x)$ is itself differentiable, this leads to the iteration formula

$$x_{i+1} = x_i - \frac{f'(x_0)}{f''(x_0)}.$$

One might wish to find a local maximum or minimum of a vector function $f : \mathbb{R}^K \to \mathbb{R}$. This requires multivariate Newton-Raphson, with generalized formula

$$\boldsymbol{x}' = \boldsymbol{x} - H^{-1}\boldsymbol{g}.$$

Here, $\boldsymbol{g}$ represents the gradient of the $f(\boldsymbol{x})$, akin to the derivative of f. H represents the *Hessian Matrix* which is a generalization of the single variable second derivative[1].

These methods are related to the popular gradient descent algorithms used in machine learning. In its simplest form, gradient descent sets $\boldsymbol{x}' = \boldsymbol{x} - \lambda\boldsymbol{g}$. By using the Hessian matrix to affect the learning rate λ, the centuries-old method is a great way to speed up a learning algorithm.

[1]Note that we are taking $\boldsymbol{x}'$ to mean *the next value of* x and not a derivative.

Sometimes inverting the hessian matrix is computationally impractical. In this case, the diagonal of the hessian matrix may be used instead[1]. However, there can be benefits to using the full Hessian in order to speed up convergence.

This technical note shows how one can invert the hessian matrix when it is a constant plug a diagonal using the matrix inversion lemma. This technique is quite common in statistical and machine learning literature[2, Appendix A.2][3, Section 4][4, p. 214][5], but there are many details left to fill in in terms of both manipulating the equations and turning them into a computational algorithm. Even though the process is a somewhat mechanical application of linear algebra, it can be quite complicated and some readers have requested a more detailed explanation of the chain of logic.

We aim to provide this here. This paper still requires a background in linear algebra and multivariate calculus, but more of the steps are included. It will help any readers with at least an undergraduate level understanding of linear algebra to fill in the missing steps.

2 Multivariate Newton's Method

Let $f : \mathbb{R}^K \to \mathbb{R}$ be an objective function for which we want to find a maximum $\arg\max_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha})$. We decide to use Newton's Method if we are satisfied with a local maximum, or if f is convex and can be shown to only have a global maximum.

We start with an initial value of $\boldsymbol{\alpha}$ and take a step as defined below to find the next value of $f(\boldsymbol{\alpha})$. We iterate until convergence. Hopefully!

Let $\boldsymbol{g}$ be the gradient[2] of f with respect to $\boldsymbol{\alpha}$. The gradient $\boldsymbol{g}(\boldsymbol{\alpha})$ can be explicitly represented as a column vector (appendix A).

$$\boldsymbol{g}(\boldsymbol{\alpha}) = \boldsymbol{g} = \nabla f(\boldsymbol{\alpha}) = \begin{bmatrix} \frac{\partial f}{\partial \alpha_0} \\ \frac{\partial f}{\partial \alpha_1} \\ \vdots \\ \frac{\partial f}{\partial \alpha_{K-1}} \end{bmatrix}.$$

Now, let the matrix H be the Hessian matrix of the function f at $\boldsymbol{\alpha}$, represented explicitly as

[2]For the duration of this paper, we will use $\boldsymbol{g}$ as shorthand for $\boldsymbol{g}(\boldsymbol{\alpha})$, and f as shorthand for $f(\boldsymbol{\alpha})$, etc. Having $(\boldsymbol{\alpha})$ peppered throughout formulas does not do us any favors with regard to readability, despite being technically correct.

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial \alpha_0^2} & \frac{\partial^2 f}{\partial \alpha_0 \partial \alpha_1} & \cdots & \frac{\partial^2 f}{\partial \alpha_0 \partial \alpha_{K-1}} \\ \frac{\partial^2 f}{\partial \alpha_1 \partial \alpha_0} & \frac{\partial^2 f}{\partial \alpha_1^2} & \cdots & \frac{\partial^2 f}{\partial \alpha_1 \partial \alpha_{K-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial \alpha_{K-1} \partial \alpha_0} & \frac{\partial^2 f}{\partial \alpha_{K-1} \partial \alpha_1} & \cdots & \frac{\partial^2 f}{\partial \alpha_{K-1}^2} \end{bmatrix}.$$

A *Newton step* for maximizing $f(\boldsymbol{\alpha})$ is defined as

$$\boldsymbol{\alpha'} = \boldsymbol{\alpha} + \Delta\boldsymbol{\alpha} = \boldsymbol{\alpha} - H^{-1}\boldsymbol{g},$$

where $\boldsymbol{\alpha'}$ is the updated vector after a single iteration, and H^{-1} is the inverse of the Hessian matrix at $\boldsymbol{\alpha}$. Our goal is to find the step size $\Delta\boldsymbol{\alpha} = -H^{-1}\boldsymbol{g}$ given by the product of the inverse Hessian matrix and the gradient vector.

3 Diagonal and Constant Matrix Case

We consider the case where the Hessian matrix H can be broken down into the sum

$$H = D + C$$

where D is a diagonal matrix and C is a constant matrix. Suppose that D is formed from the vector $\boldsymbol{d} = [d_0, d_1, \ldots, d_{K-1}]^\top$ (appendix B) so that

$$D = \operatorname{diag}(\boldsymbol{d}) = \begin{bmatrix} d_0 & 0 & \cdots & 0 \\ 0 & d_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_{K-1} \end{bmatrix}$$

Let $\mathbf{1}_K$ be a vector of 1s with K entries:

$$\mathbf{1}_K = [1, 1, \ldots, 1]^\top.$$

The constant matrix C is defined as the *outer product* (appendix A) of $\mathbf{1}_K$ and itself, scaled by the constant c:

$$C = c \cdot \mathbf{1}_K \mathbf{1}_K^\top = c \cdot \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} = \begin{bmatrix} c & c & \cdots & c \\ c & c & \cdots & c \\ \vdots & \vdots & \ddots & \vdots \\ c & c & \cdots & c \end{bmatrix}.$$

4 Solution Using the Matrix Inversion Lemma

The Woodbury matrix inversion lemma[6] is given as follows:

Let $A \in \mathbb{R}^{n\times n}$ be an invertible matrix, $U \in \mathbb{R}^{n\times k}$, $V \in \mathbb{R}^{k\times n}$, and $C \in \mathbb{R}^{k\times k}$. If $C^{-1} + VA^{-1}U$ is invertible, then the inverse of the matrix $A + UCV$ can be expressed as:

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}.$$

In our circumstances, we can use a special case of this lemma known as the *Sherman-Morrison formula*[7]:

Let $A \in \mathbb{R}^{n\times n}$ be an invertible matrix, with $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$.

$$\left(A + \boldsymbol{u}\boldsymbol{v}^\top\right)^{-1} = A^{-1} - \frac{A^{-1}\boldsymbol{u}\boldsymbol{v}^\top A^{-1}}{1 + \boldsymbol{v}^\top A^{-1}\boldsymbol{u}} \tag{1}$$

In order to apply this to our question, we make the correspondence

$$\begin{pmatrix} A \\ \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} \to \begin{pmatrix} D \\ c\mathbf{1}_K \\ \mathbf{1}_K \end{pmatrix}.$$

Then, we get the following result by plugging this correspondence into equation 1.

$$H^{-1} = \left(D + c\mathbf{1}_K\mathbf{1}_K^\top\right)^{-1} = D^{-1} - \frac{D^{-1}c\mathbf{1}_K\mathbf{1}_K^\top D^{-1}}{1 + \mathbf{1}_K^\top D^{-1}c\mathbf{1}_K} = D^{-1} - \frac{D^{-1}\mathbf{1}_K\mathbf{1}_K^\top D^{-1}}{c^{-1} + \mathbf{1}_K^\top D^{-1}\mathbf{1}_K}$$

Fortunately, the inverse of a diagonal matrix D can be computed by simply taking the multiplicative inverses of the diagonal elements (appendix B). Therefore, we define $\boldsymbol{d}^{-1}$ as the vector of values along the diagonal of D^{-1}.

$$\boldsymbol{d}^{-1} = [d_0^{-1}, d_1^{-1}, \ldots, d_{K-1}^{-1}]^\top \qquad D^{-1} = \operatorname{diag}(\boldsymbol{d}^{-1})$$

Notice the following interactions between D^{-1} and $\mathbf{1}_K$:

$$\boldsymbol{D}^{-1}\mathbf{1}_K = \boldsymbol{d}^{-1} \qquad \mathbf{1}_K^\top\boldsymbol{D}^{-1} = \boldsymbol{d}^{-1\,T} \qquad \mathbf{1}_K D^{-1}\mathbf{1}_K^\top = \sum_{k=1}^{K-1} d_k^{-1}$$

Using these equivalences, we can simplify our formula to $H^{-1} = D^{-1} - \frac{\boldsymbol{d}^{-1}\boldsymbol{d}^{-1\,T}}{c^{-1} + \sum_{k=1}^{K-1} d_k^{-1}}$.

We now let $Z = c^{-1} + \sum_{k=1}^{K-1} d_k^{-1}$ to get $H^{-1} = D^{-1} - \frac{\boldsymbol{d}^{-1}\boldsymbol{d}^{-1\,T}}{Z}$. To get the next Newton step, we multiply this by $\boldsymbol{g}$ to get the following.

$$\Delta\boldsymbol{\alpha} = -H^{-1}\boldsymbol{g} = -D^{-1}\boldsymbol{g} + \frac{\boldsymbol{d}^{-1}\boldsymbol{d}^{-1\,T}}{Z}\boldsymbol{g} = -D^{-1}\boldsymbol{g} + \frac{\boldsymbol{d}^{-1}\boldsymbol{d}^{-1\,T}\boldsymbol{g}}{Z}$$

Now find a formula for a single component k of $\Delta\boldsymbol{\alpha}$. (see appendix B for manipulation of $D^{-1}\boldsymbol{g}$).

$$(\Delta\boldsymbol{\alpha})_k = -\left(D^{-1}\boldsymbol{g}\right)_k + \left(\frac{\boldsymbol{d}^{-1}\boldsymbol{d}^{-1\,T}\boldsymbol{g}}{Z}\right)_k = -\frac{g_k}{d_k} + \left(\frac{1}{Z}\right)\frac{\boldsymbol{d}^{-1}\cdot\boldsymbol{g}}{d_k}$$

Let $S = \boldsymbol{d}^{-1}\cdot\boldsymbol{g}$ to get

$$(\Delta\boldsymbol{\alpha})_k = -\frac{g_k}{d_k} + \left(\frac{S}{Z}\right)\frac{1}{d_k} = \frac{S\cdot Z^{-1} - g_k}{d_k}$$

This process directly translates to algorithm 1.

Algorithm 1 Algorithm for a Newton Step

```
function STEP($\mathbf{g} \in \mathbb{R}^K, \mathbf{d} \in \mathbb{R}^K, c \in \mathbb{R}$)
    $S \leftarrow 0$
    for $k = 0$ to $K - 1$ do
        $S \leftarrow S + \mathbf{g}[k]/\mathbf{d}[k]$
    $Z \leftarrow 1/c$
    for $k = 0$ to $K - 1$ do
        $Z \leftarrow Z + 1/\mathbf{d}[k]$
    $\delta \in \mathbb{R}^K$
    for $k = 0$ to $K - 1$ do
        $\delta[k] \leftarrow (S/Z - \mathbf{g}[k])/\mathbf{d}[k]$
    return $\delta$
```

5 Newton Step in Log Space

Suppose that the values of $\boldsymbol{\alpha}$ are all required to be positive. Sometimes, the Newton step will cause an $\boldsymbol{\alpha}_k$ to go negative, which is out of bounds. If this is the case, we can instead find a step on a function applied to the vector $\boldsymbol{\beta}$ where

$$\boldsymbol{\beta} = [\ln(\alpha_0), \ln(\alpha_1), \ldots, \ln(\alpha_{K-1})]^\top.$$

We now say that the vector $\boldsymbol{\beta}$ lives in *log space*, meaning that every possible value will lead to a vector of only positive elements $\boldsymbol{\alpha}$ to be passed into f. We now have a new objective function

$$f^*(\boldsymbol{\beta}) = f(\boldsymbol{\alpha})$$

.

We noticed that f^* and f are actually the same function on transformed inputs, so when looked at from the perspective of these being values dependent on the inputs, $f^* = f$ with each being uniquely determined by either $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$. We also note that the derivative of a component of $\boldsymbol{\alpha}$ with respect to the same component of $\boldsymbol{\beta}$:

$$\frac{\partial \alpha_k}{\partial \beta_k} = \frac{\partial}{\partial \beta_k} e^{\beta_k} = e^{\beta_k} = \alpha_k$$

We will find a log-space gradient $\boldsymbol{g}^*$ to distinguish it from the old gradient. The gradient is now

$$g_k^* = \frac{\partial}{\partial \beta_k} f = \frac{\partial f}{\partial \alpha_k}\frac{\partial \alpha_k}{\partial \beta_k} = g_k \alpha_k.$$

To find the log-space Hessian H^*, we look at a single component $h_{i,j}^*$.

$$h_{i,j}^* = \frac{\partial}{\partial \beta_i}\frac{\partial}{\partial \beta_j} f = \frac{\partial}{\partial \beta_i} g_j \alpha_j = h_{i,j}\alpha_j\alpha_i + g_j \frac{\partial \alpha_j}{\partial \beta_i} = h_{i,j}(\alpha)\alpha_j\alpha_i + g_j \mathbb{I}_{i,j}\alpha_j$$

Note that for the old hessian (H), we know that $h_{i,j} = c + \mathbb{I}_{i,j}\boldsymbol{d}_i$. $\mathbb{I}_{i,j}$ is the indicator function which is equal to 1 when $i = j$ and 0 otherwise. This forces the $\boldsymbol{d}_i$ terms to only appear on the diagonal[3] With the ability to break down $h_{i,j}$, our equation is now as follows

$$h_{i,j}^* = \left(c + \mathbb{I}_{i,j} d_i\right)\alpha_j\alpha_i + g_j \mathbb{I}_{i,j}\alpha_i = c\alpha_j\alpha_i + \mathbb{I}_{i,j}\left(\alpha_i^2 d_i + g_i\alpha_i\right)$$

For convenience, define the vector $\boldsymbol{x} \in \mathbb{R}^K$ with $x_i = \alpha_i d_i + g_i$. This simplifies the hessian formula to

$$h_{i,j}^* = c\alpha_j\alpha_i + \mathbb{I}_{i,j}\alpha_i x_i$$

Note that instead of a constant matrix, we have a constant c multiplied by the outer product of $\boldsymbol{\alpha}$ with itself. The outer product is illustrated here, and note how it generates the term $\alpha_j\alpha_i$.

$$\boldsymbol{\alpha}\boldsymbol{\alpha}^\top = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{K-1} \end{bmatrix} \begin{bmatrix} \alpha_0 & \alpha_1 & \cdots & \alpha_{K-1} \end{bmatrix} = \begin{bmatrix} \alpha_0^2 & \alpha_0\alpha_1 & \cdots & \alpha_0\alpha_{K-1} \\ \alpha_1\alpha_0 & \alpha_1^2 & \cdots & \alpha_1\alpha_{K-1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{K-1}\alpha_0 & \alpha_{K-1}\alpha_1 & \cdots & \alpha_{K-1}^2 \end{bmatrix}.$$

Let $D^* = \operatorname{diag}(\boldsymbol{\alpha})\operatorname{diag}(\boldsymbol{x})$. We can now piece together the full version of the hessian as:

$$H^* = c\boldsymbol{\alpha}\boldsymbol{\alpha}^\top + \operatorname{diag}(\boldsymbol{\alpha})\operatorname{diag}(\boldsymbol{x}) = D^* + \boldsymbol{\alpha}c\boldsymbol{\alpha}^\top$$

[3]We could equivalently write $\boldsymbol{d}_j$ as $i = j$, but let's use the first subscript as a standard.

This still fits with the Sherman-Morrison formula and we get a new correspondence

$$\begin{pmatrix} A \\ \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} \to \begin{pmatrix} D^* \\ c\boldsymbol{\alpha} \\ \boldsymbol{\alpha} \end{pmatrix}.$$

We can now invert the log-space hessian by plugging these values in to equation 1.

$$H^{*-1} = \left(D^* + c\boldsymbol{\alpha}\boldsymbol{\alpha}^\top\right)^{-1} = D^{*-1} - \frac{D^{*-1}c\boldsymbol{\alpha}\boldsymbol{\alpha}^\top D^{*-1}}{1 + \boldsymbol{\alpha}^\top D^{*-1}c\boldsymbol{\alpha}}$$

To get a Newton step, we multiply this by the gradient $\boldsymbol{g}^*$ to get

$$\Delta\boldsymbol{\beta} = -H^{*-1}\boldsymbol{g}^* = -D^{*-1}\boldsymbol{g}^* + \frac{D^{*-1}c\boldsymbol{\alpha}\boldsymbol{\alpha}^\top D^{*-1}\boldsymbol{g}^*}{1 + \boldsymbol{\alpha}^\top D^{*-1}c\boldsymbol{\alpha}}.$$

Now to find a single component k:

$$(\Delta\boldsymbol{\beta})_k = -\frac{g_k^*}{\alpha_k x_k} + \frac{\left[D^{*-1}\boldsymbol{\alpha}\boldsymbol{\alpha}^\top D^{*-1}\boldsymbol{g}^*\right]_k}{(c^{-1} + \boldsymbol{\alpha}^\top D^{*-1}\boldsymbol{\alpha})}$$

The pre-multiplication by D^{*-1} in the numerator will simply divide each term by $\boldsymbol{\alpha}_k\boldsymbol{x}_k$, allowing us to further simplify to:

$$(\Delta\boldsymbol{\beta})_k = -\frac{g_k^*}{\alpha_k x_k} + \frac{\left[\boldsymbol{\alpha}\boldsymbol{\alpha}^\top D^{*-1}\boldsymbol{g}^*\right]_k}{\alpha_k x_k(c^{-1} + \boldsymbol{\alpha}^\top D^{*-1}\boldsymbol{\alpha})}$$

Finally, note that $\boldsymbol{\alpha}^\top D^{*-1}\boldsymbol{g}^*$ is a scalar equivalent to the dot product $\boldsymbol{\alpha} \cdot D^{*-1}\boldsymbol{g}^*$. This allows us to pull this dot product out, and then extract the field from the first $\boldsymbol{\alpha}$ like so:

$$(\Delta\boldsymbol{\beta})_k = -\frac{g_k^*}{\alpha_k x_k} + \frac{\alpha_k\left[\boldsymbol{\alpha}^\top D^{*-1}\boldsymbol{g}^*\right]}{\alpha_k x_k(c^{-1} + \boldsymbol{\alpha}^\top D^{*-1}\boldsymbol{\alpha})} = -\frac{g_k^*}{\alpha_k x_k} + \frac{\boldsymbol{\alpha}^\top D^{*-1}\boldsymbol{g}^*}{x_k(c^{-1} + \boldsymbol{\alpha}^\top D^{*-1}\boldsymbol{\alpha})}$$

Now we make some replacements for the scalar terms to make this more readable. Start with Z for the term in the denominator.

$$Z = c^{-1} + \boldsymbol{\alpha}^\top D^{*-1}\boldsymbol{\alpha} = \frac{1}{c} + \sum_{k=0}^{K-1}\frac{\alpha_k^2}{\alpha_k x_k} = \frac{1}{c} + \sum_{k=0}^{K-1}\frac{\alpha_k}{x_k}$$

Next find a formula for S, the term in the numerator.

$$S = \boldsymbol{\alpha}^\top D^{*-1}\boldsymbol{g}^* = \sum_{k=0}^{K-1} \frac{\alpha_k g_k^*}{\alpha_k x_k} = \sum_{k=0}^{K-1} \frac{g_k^*}{x_k} = \sum_{k=0}^{K-1} \frac{\alpha_k g_k}{x_k}$$

Now put it together for a final, simplified form.

$$(\Delta\boldsymbol{\beta})_k = -\frac{g_k^*}{\alpha_k x_k} + \frac{S}{x_k Z} = \frac{1}{x_k}\left(-\frac{g_k^*}{\alpha_k} + \frac{S}{Z}\right) = \frac{1}{x_k}\left(\frac{S}{Z} - g_k\right)$$

Finally, this translates into the following algorithm:

Algorithm 2 Algorithm for a Newton Step In Log Space

```
function StepInLogSpace($\alpha \in \mathbb{R}^K, \mathbf{g} \in \mathbb{R}^K, \mathbf{d} \in \mathbb{R}^K, c \in \mathbb{R}$)
    $x \in \mathbb{R}^K$
    for $k = 0$ to $K - 1$ do
        $x_k \leftarrow \mathbf{g}[k] + \alpha[k] \cdot \mathbf{d}[k]$
    $Z \leftarrow 1/c$
    for $k = 0$ to $K - 1$ do
        $Z \leftarrow Z + \alpha[\mathbf{k}]/\mathbf{x}[k]$
    $S \leftarrow 0$
    for $k = 0$ to $K - 1$ do
        $S \leftarrow S + \alpha[k] \cdot \mathbf{g}[k]/\mathbf{x}[k]$
    $\delta \in \mathbb{R}^K$
    for $k = 0$ to $K - 1$ do
        $\delta[k] \leftarrow (S/Z - \mathbf{g}[k])/\mathbf{x}[k]$
    return $\delta$
```

Given that this is a Newton step on $\boldsymbol{\beta}$, we must take care to understand how it affects $\boldsymbol{\alpha}$. If the next value for $\boldsymbol{\beta}$ is $\boldsymbol{\beta} + \Delta\boldsymbol{\beta}$, then the next value of $\boldsymbol{\alpha}$ is actually

$$\alpha_k' = e^{\beta_k'} = e^{\beta_k + (\Delta\boldsymbol{\beta})_k} = e^{\beta_k} \cdot e^{(\Delta\boldsymbol{\beta})_k} = \alpha_k \cdot e^{(\Delta\boldsymbol{\beta})_k}$$

This formula allows us to use algorithm 2 in order to find $\Delta\boldsymbol{\beta}$ which could then be used to find the next set of positive components of $\boldsymbol{\alpha}$.

Appendices

A Inner and Outer Products

Unless noted otherwise, a vector is assumed to be a *column vector*, meaning that its components are arranged as a single column with multiple rows. For a vector $\boldsymbol{a} \in \mathbb{R}^K$, the column vector form is:

$$\boldsymbol{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{k-1} \end{bmatrix}.$$

A row vector has a single row and multiple columns. It is the transpose of a column vector give $\boldsymbol{a}$ as follows.

$$\boldsymbol{a}^\top = \begin{bmatrix} a_0 & a_1 & \cdots & a_{K-1} \end{bmatrix}.$$

The dot product of two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^K$ is defined as the sum of the products of their corresponding components.

$$\boldsymbol{a} \cdot \boldsymbol{b} = \sum_{i=0}^{K-1} a_i b_i.$$

The matrix multiplication of a row vector and a column vector results in a 1×1 matrix (or a *scalar*) whose single component is the dot product of the two vectors. Given a row vector $\boldsymbol{a}^\top$ and a column vector $\boldsymbol{b}$ both of dimension K (or rather dimensions $1 \times K$ and $K \times 1$), their matrix multiplication is:

$$\boldsymbol{a}^\top \boldsymbol{b} = \begin{bmatrix} a_0 & a_1 & \cdots & a_{K-1} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{K-1} \end{bmatrix} = \sum_{i=0}^{K-1} a_i b_i = \boldsymbol{a} \cdot \boldsymbol{b}.$$

Now if instead a column vector ($N \times 1$) were multiplied by a row vector ($1 \times M$), the resulting $N \times M$ matrix is the *outer product.*

$$\boldsymbol{a}\boldsymbol{b}^\top = \boldsymbol{a} \otimes \boldsymbol{b} = \begin{bmatrix} a_0 b_0 & a_0 b_1 & \cdots & a_0 b_{M-1} \\ a_1 b_0 & a_1 b_1 & \cdots & a_1 b_{M-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N-1} b_0 & a_{N-1} b_1 & \cdots & a_{N-1} b_{M-1} \end{bmatrix}.$$

B Diagonal Matrices

A diagonal matrix is a type of matrix in which the entries outside the main diagonal are all zero. The main diagonal entries can be represented by a vector $\boldsymbol{d} \in \mathbb{R}^K$.

The operator $\texttt{diag}(\boldsymbol{d})$ transforms the vector $\boldsymbol{d}$ into a diagonal matrix D:

$$D = \texttt{diag}(\boldsymbol{d}) = \begin{bmatrix} d_0 & 0 & \cdots & 0 \\ 0 & d_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_{K-1} \end{bmatrix}.$$

The inverse of a diagonal matrix D is also a diagonal matrix. The entries of the inverse matrix are the multiplicative inverses of the entries of D:

$$D^{-1} = \texttt{diag}(\boldsymbol{d}^{-1}) = \begin{bmatrix} d_0^{-1} & 0 & \cdots & 0 \\ 0 & d_1^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_{K-1}^{-1} \end{bmatrix}.$$

When a diagonal matrix multiplies a vector, the operation is equivalent to element-wise multiplication (or *Hadamard product* $\odot$) between the vector and the diagonal entries of the matrix:

$$D\boldsymbol{v} = \texttt{diag}(\boldsymbol{d})\boldsymbol{v} = \boldsymbol{d} \odot \boldsymbol{v} = \begin{bmatrix} d_0 v_0 \\ d_1 v_1 \\ \vdots \\ d_{K-1} v_{K-1} \end{bmatrix}.$$

References

[1] Becker, S and Lecun, Y., (1988) Improving the convergence of backpropagation learning with second-order methods. In Proceedings of the 1988 Connectionist Models Summer School, San Mateo, D. Touretzky, G. Hinton, and T. Sejnowski, Eds. Morgan Kaufmann, 1989, pp. 29–37.

[2] Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. the Journal of machine Learning research, 3, 993-1022.

[3] Sklar, M. (2014). Fast MLE computation for the Dirichlet multinomial. arXiv preprint arXiv:1405.0099.

[4] Ng, K. W., Tian, G. L., & Tang, M. L. (2011). Dirichlet and Related Distributions: Theory, Methods and Applications (Vol. 888). John Wiley & Sons.

[5] Minka, T. (2000). Estimating a Dirichlet distribution. Technical report, M.I.T., 2000.

[6] Woodbury, M. A. (1950). Inverting modified matrices. Memorandum report 42, Statistical Research Group. Princeton University.

[7] Sherman, Jack, and Winifred J. Morrison. (1950). Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. The Annals of Mathematical Statistics 21.1 (1950): 124-127.