COMS 4705 requires some mathematical comfort with vectors, matrix algebra, probability distributions, a bit of differential calculus, and some ideas from optimization. You may be able to fill in some or all of these gaps by learning as we go. This note is to aid in filling in gaps, and provide a few of the "tricks" or bits of nice math that may come up. This note should also help demystify some of the language used in the course if you're not yet comfortable with these topics.

This document does not intend to be exhaustive. Some properties are left undefined, and many important things are missing. We're aiming to strike a balance of concision and coverage. If you have comments, suggested additions, suggested removals, or (especially) bug reports, please send them to jh5020@columbia.edu.

Vectors and Matrix Algebra

Vectors

A vector $\mathbf{v}$ in space $\mathbb{R}^d$ is said to be d-dimensional. We'll sometimes say "real-valued vectors" to refer to such vectors, as their dimensions (each individual v_i, where $i \in \{1, \ldots, d\}$) are in the reals, $\mathbb{R}^d$. However, we work with computers and these vectors never really contain reals; sometimes we use a very small finite field, like using 32 bits, or 16, or 8, or even 4 to specify the possible values each dimension can take. This will sometimes be important, but in our notation, we'll still say $\mathbb{R}^d$.

A dot product between vectors produces a scalar quantity. Let $\mathbf{v}$ and $\mathbf{u}$ be in $\mathbb{R}^d$. Then their dot product is

$$\mathbf{u}^\top \mathbf{v} = \sum_{i=1}^{d} u_i v_i \tag{1}$$

Matrices

A matrix $W \in \mathbb{R}^{m \times n}$ is said to be an m *by* n matrix. A matrix is said to specify a linear mapping. By linear, we mean things like $y = mx + b$. In the case of W as we've defined it, it's a mapping from vectors in $\mathbb{R}^n$ to vectors in $\mathbb{R}^m$.

Note that W is m vectors of n dimensions each, and also can be seen as n vectors of m dimensions each. As such, we can *index into* W, as $W_{i,:}$, to refer to the i^{th} n-dimensional vector, or as $W_{:,i}$ to refer to the i^{th} m-dimensional vector. As such, I could for example compute a dot product $W_{i,:}^\top W_{j,:}$ between two n-dimensional vectors. I could also index into W as W_{ij} to refer to the scalar value there.

Let $\mathbf{a}$ be a vector in $\mathbb{R}^n$. In a **matrix-vector product**, we have

$$(W\mathbf{a})_i = W_{i,:}^\top \mathbf{a} \tag{2}$$

$$\forall i \in \{1, \ldots, m\} \tag{3}$$

Note here that I have parentheses around $W\mathbf{a}$ to clarify that the index i indexes into the result of the matrix-vector product. Matrix-vector products (and thus matrix-matrix products) are a bunch of dot products.

A matrix-matrix product between $W \in \mathbb{R}^{m \times n}$ and $G \in \mathbb{R}^{n \times d}$ is a matrix in $\mathbb{R}^{m \times d}$. This matrix-matrix product is possible because the dimension "in the middle" is the same, n. The remaining dimensions, you'll notice, are the first dimensionality of W (m) and

the second dimensionality of G (d). We have

$$(WG)_{ik} = W_{i,:}^\top G_{:,k} \tag{4}$$
$$i \in \{1, \ldots, m\} \tag{5}$$
$$k \in \{1, \ldots, d\} \tag{6}$$

Some useful vector and matrix properties

Two vectors $\mathbf{u}$ and $\mathbf{v}$ in the same dimensionality $\mathbb{R}^d$ are said to be *orthogonal* if

$$\mathbf{u}^\top \mathbf{v} = 0. \tag{7}$$

We sometimes denote this as $\mathbf{u} \perp \mathbf{v}$.

Norms

The L_2 norm of a vector $\mathbf{v} \in \mathbb{R}^d$ is

$$\|\mathbf{v}\|_2 = \sqrt{\sum_{i=1}^{d} v_i^2} \tag{8}$$

We'll also often just call this the "norm" of the vector, without specifying L_2. There are, however, many other norms, for example, the L_1 norm,

$$\|\mathbf{v}\|_1 = \sum_{i=1}^{d} |v_i|. \tag{9}$$

Another fun one is the infinity norm,

$$\|\mathbf{v}\|_\infty = \max_{i=1}^{d} |v_i|. \tag{10}$$

There are some fine matrix norms as well; one is the Frobenius norm, which is like L_2. For $W \in \mathbb{R}^{m \times n}$, we have

$$\|W\|_F = \sqrt{\sum_{i,j} W_{ij}^2} \tag{11}$$

Outer products

The outer product $\mathbf{u}\mathbf{v}^\top$ between two vectors $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{v} \in \mathbb{R}^n$ is the matrix

$$(\mathbf{u}\mathbf{v}^\top)_{ij} = u_i v_j \tag{12}$$

Note that $\mathbf{u}\mathbf{v}^\top$ is in $\mathbb{R}^{m \times n}$. This is an interesting matrix. You can think of it as n copies of the vector $\mathbf{u}$ stacked together, each one weighted by some scalar v_j. You can also think of it as m copies of the vector $\mathbf{v}$ stacked together, each weighted by some scalar u_i. But what's best about this matrix, and the notation we've used for it, is the transparent semantics. It looks like $\mathbf{u}\mathbf{v}^\top$, and you can treat it as such in a matrix-vector product. Consider:

$$(\mathbf{u}\mathbf{v}^\top)\mathbf{v} = \mathbf{u}(\mathbf{v}^\top \mathbf{v}) = \mathbf{u}\|\mathbf{v}\|_2^2 \tag{13}$$

So, here I'm just multiplying the vector $\mathbf{u}$ with the scalar $\|\mathbf{v}\|_2^2$, that is, the squared norm of $\mathbf{v}$. This transparent semantics is nice in the sense that this matrix intuitively maps the extent of $\mathbf{v}$-ness of a vector $\mathbf{w}$ (through the dot product $\mathbf{v}^\top\mathbf{w}$) to a resulting amount of $\mathbf{u}$-ness. Put another way, it takes the extent to which a vector $\mathbf{w}$ aligned with $\mathbf{v}$ and makes the resulting vector align with $\mathbf{u}$ to the same amount. Also note that if $\mathbf{w} \perp \mathbf{v}$, then we have

$$(\mathbf{u}\mathbf{v}^\top)\mathbf{w} = \mathbf{u}(\mathbf{v}^\top\mathbf{w}) = \mathbf{u} \times 0 = \mathbf{0} \tag{14}$$

Linear independence

A vector $\mathbf{u}$ is linearly dependent on a set of vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$, all in $\mathbb{R}^d$, if there exist scalars $c_1, \ldots, c_m$ such that

$$\mathbf{u} = \sum_{i=1}^{m} c_i \mathbf{v}_i \tag{15}$$

The vector $\mathbf{u}$ is said to be linearly independent from that set if no such scalars exist. One way to put it is that the set $\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ are like ingredients in an additive soup, or building blocks, and linear independence states that it's impossible to construct $\mathbf{u}$ from any possible additive combination of those building blocks or soup ingredients.

Matrix rank

The rank of a matrix W is the minimum number k such that we can represent W exactly as the sum of k outer products:

$$W = \sum_{i=1}^{k} \mathbf{u}^{(i)}{\mathbf{v}^{(i)}}^\top \tag{16}$$

Note here how I've used $\mathbf{u}^{(i)}$ to specify the i^{th} vector out of some list of vectors, not putting the index at the bottom so as not to suggest I'm indexing into the vector to denote a scalar. Also note that thus an outer product $\mathbf{u}\mathbf{v}^\top$ has rank 1, since it can be represented by the sum of just itself.

The rank is also (and is usually introduced as) the number of linearly independent m-dimensional vectors (or equivalently, the number of linearly independent n-dimensional vectors) that make up W.[1] However, I like the outer product form more.

Representing W as its sum of outer products when discussing rank is useful for a number of reasons. For now, the only one I'll mention is it makes intuitive to me the fact that the rank of W is often used as a **measure of the complexity of the map defined by** W. That is, if $W = \mathbf{u}\mathbf{v}^\top$, then the function is simple – it maps the affinity with $\mathbf{v}$ to a corresponding affinity with $\mathbf{u}$, and maps everything else to 0. If $W = \mathbf{u}^{(1)}{\mathbf{v}^{(1)}}^\top + \mathbf{u}^{(2)}{\mathbf{v}^{(2)}}^\top$, then the function defined by W is sort of doing two things at once. Nice!

Eigenvectors and eigendecomposition

Much as with matrix rank, eigenvectors (and singular vectors) are going to be useful more for getting a feeling for the properties of a matrix than for their definitional properties.

[1] Sometimes these vectors are called "row vectors" and "column vectors" correspondingly, but I always found that confusing, especially as we venture into more complex matrix-like objects, so I note it here only so you'll not be lost if you see those terms.

Here's a definition: an eigenvector of matrix $W \in \mathbb{R}^{d \times d}$ is a vector $\mathbf{v}$ such that it is a unit-norm vector ($\|\mathbf{v}\| = 1$), and that

$$W\mathbf{v} = \sigma\mathbf{v}. \tag{17}$$

The scalar value σ is the eigenvector's corresponding eigenvalue. So, eigenvectors of W are vectors that are only *scaled*, not changed in direction. As such, it is only defined for square matrices. Also, the eigenvectors of W are mutually orthogonal.

When a $d \times d$ matrix has d eigenvectors (this is always true when the matrix is symmetric), it can be represented by its **eigendecomposition**. Let $V \in \mathbb{R}^{d \times d}$ be a matrix of the d eigenvectors, sorted in order of decreasing absolute value of their eigenvalues. Let Σ be the diagonal matrix of just the sorted eigenvalues. Then we can represent W as

$$W = V\Sigma V^\top \tag{18}$$

This doesn't seem terribly enlightening to me; I prefer to write it out as the following sum:

$$W = \sum_{i=1}^{d} \sigma_i V_i V_i^\top, \tag{19}$$

where σ_i is the i^{th} eigenvalue. So, just like in our matrix rank section, we've written the matrix as a sum over simple outer products. But additionally here, intuitively, because we've sorted the eigenvalues in decreasing absolute value, taking only the first k of d values of this sum is like taking the **most important** components of the matrix. This is true more formally; taking the first k components of this sum computes the **best rank-k approximation to** W under the Frobenius (L_2-like) distance. That is,

$$\min_{\hat{W}|\text{rank}(\hat{W})=k} \|W - \hat{W}\|_F = \sum_{i=1}^{k} \sigma_i V_i V_i^\top \tag{20}$$

So intuitively, it's useful to think of the eigendecomposition of a matrix as telling us a decomposition of the function computed by the matrix in decreasing order of importance. If W has n real non-zero eigenvalues, it will be full-rank. But, if the sorted eigenvalues—also called the spectrum—quickly *decay*, that is, maybe the first is $\sigma_1 = 1$ and then $\sigma_2 = 0.5$ and then $\sigma_3 = 0.25$... and in general $\sigma_i = \frac{1}{2^i}$, then the outer product matrix corresponding to, say, σ_{10} has a very small contribution to the matrix! (Note that this is because all the eigenvectors are unit norm, so the outer products $V_i V_i^\top$ can't incorporate a large scaling factor themselves.) So, this matrix, despite being full rank, doesn't *feel* like a full-rank matrix, since it's well-approximated by lower-rank matrices.

Singular vectors and singular value decomposition

Some square matrices over the reals $\mathbb{R}$ don't have n eigenvectors[2]. Many matrices are not square.[3] In these cases, we can't use the intuitions we've built about the eigendecomposition to reason about the properties of matrices. However, we can do something basically just as good for these intuitive purposes, leveraging the connnected idea of **singular vectors**.

[2] Though if you define the matrix over the complex numbers $\mathbb{C}$, they do – because the corresponding eigenvalues are complex. We will not use this.

[3] Citation needed.

Let $W \in \mathbb{R}^{m \times d}$ (so this time it's not necessarily square.) A pair of left- and right-singular vectors $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{v} \in \mathbb{R}^d$ are unit-norm vectors such that

$$W\mathbf{v} = \sigma\mathbf{u} \tag{21}$$

$$W^\top \mathbf{u} = \sigma\mathbf{v}. \tag{22}$$

As before with eigenvectors, we'll be much more interested in the properties of the **singular value decomposition** of a matrix. Intuitively, let $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{d \times d}$ be the matrices representing the left- and right-singular vectors of a matrix, and let $\Sigma \in \mathbb{R}^{m \times d}$ be the diagonal(ish)[4] matrix of singular values. Then

$$W = U\Sigma V^\top \tag{23}$$

Writing this out again as a weighted sum of outer product matrices, we get

$$W = \sum_{i=1}^{\min(m,d)} \sigma_i U_i V_i^\top \tag{24}$$

Note how, much as in our discussion of rank, we've written out W as a weighted sum of simple matrices—simple functions—mapping some u to some v. The weighting σ_i—the singular value—is a description of the importance of that component of the map (in an L_2 sense.) And as in our discussion of the eigendecomposition, we have that the best rank-k approximation to W (as measured by L_2 distance) is that given by the sum over the first k terms in the singular value decomposition.

Higher-order tensors.

A tensor $T \in \mathbb{R}^{m \times n \times d}$ is called a 3-axis tensor. We can specify tensors with as many axes as we'd like. Those we're already familiar with are 2-axis tensors (matrices), 1-axis tensors (vectors) and 0-axis tensors (scalars.) Tensors are a bit hard for our brains to understand. But one nice hack to try with yourself is to take your k-axis tensor, and say "oh this is just a bunch of $(k-1)$-axis tensors stacked together!" and if $(k-1)$-axis tensors are easy to understand, great! And if not, just repeat this until you hit some number of axes you're comfortable with. This mostly isn't a joke; in any tensor-tensor operation, it does eventually boil down to a lot of dot products.

A tensor-vector product between $T \in \mathbb{R}^{m \times n \times d}$ and $\mathbf{v} \in \mathbb{R}^d$ is

$$(T\mathbf{v})_{ij} = T_{ij,:}^\top \mathbf{v} \tag{25}$$

Note that we've lost the last axis of T to its multiplication with $\mathbf{v}$, so $(T\mathbf{v}) \in \mathbb{R}^{m \times n}$ is a matrix (or, 2-axis tensor.)

A tensor-matrix product between $T \in \mathbb{R}^{m \times n \times d}$ and $W \in \mathbb{R}^{d \times g}$ is

$$(TW)_{ij\ell} = T_{i,j,:}^\top W_{:,\ell} \quad \ell \in \{1, \ldots, g\} \tag{26}$$

Note here again how shapes of the tensors help us see that this is possible. T is of shape $m \times n \times d$ and W is of shape $d \times g$. When we line up the shapes in the order of the multiplication, we see that the d-dimension axes are next to each other.

[4] Diagonal matrices are square and non-zero only along the diagonal (W_{ij} where $i = j$). This matrix Σ has non-zero entries on what *looks* like the diagonal if you were to lop off a bunch of all-zero values of either rows or columns depending on whether m or d is larger.

Probability Distributions

Probability distributions express uncertainty over the value of a variable. A discrete distribution specifies a countable[5] space $\mathcal{X}$ of possible outcomes. A probability distribuion p over countable space $\mathcal{X}$ defines a function from elements $x \in \mathcal{X}$ to scalars, such that

$$\forall x \in \mathcal{X}, p(x) \geq 0 \qquad \text{non-negativity} \tag{27}$$

$$\sum_{x \in \mathcal{X}} p(x) = 1 \qquad \text{normalization} \tag{28}$$

So, every element gets a probability. All probabilities are non-negative. The sum over all probabilities for elements in the set is 1.

We often say something like $x \sim p(x)$ to denote that x refers to a random sample from the distribution p.

Expectations

Taking the expectation of a quantity over a random variable means taking a weighted average of what that quantity would be for every value of the random variable, where the weights in the average are specified by the probabilities of the distribution of the random variable. So, if I have a space of strings $\{\texttt{a}, \texttt{aaa}, \texttt{ab}\}$ and a random variable distributed according to probability distribution $\{0.25, 0.5, 0.25\}$, and I define a function length(x) which is the number of letters in the string, then the expected number of letters in x is

$$0.25 \times 1 + 0.5 \times 3 + 0.25 \times 2 = 2.25 \tag{29}$$

More generally, let $x \sim p(x)$ from space $\mathcal{X}$. Let $f : x \mapsto f(x)$ be a function whose range (output space) is some space with addition defined on it. Then the expectation of $f(x)$ over p is

$$E_{x \sim p(x)}\left[f(x)\right] = \sum_{x \in \mathcal{X}} p(x) f(x) \tag{30}$$

Entropy

The entropy of a probability distribution is a statement of the uncertainty one should have over which element in the set a random sample from said distribution will be.

The entropy of a discrete distribution p over space $\mathcal{X}$ is

$$H(p) = -\sum_{x \in \mathcal{X}} p(x) \log p(x) \tag{31}$$

One way to think of the entropy is the (sometimes fractional) number of unbiased coin flips' worth of uncertainty there is in the value of a random variable. If I flip a coin once, I have a probability distribution $\{0.5, 0.5\}$ over $\{\text{heads}, \text{tails}\}$. The entropy is

$$-(0.5 \log(0.5) + 0.5 \log(0.5)) = -(0.5 \times (-1) + 0.5 \times (-1)) = 1 \tag{32}$$

This is assuming the base of the log is 2. Often we'll take log-base e instead, but the intuition is the same; it's just hard to visualize an e-sided coin.

[5] If you don't know what countable means, don't worry about it. Some of the time, we'll mean "finite". When the space is infinitely large but countable, it usually means we're referring to infinitely many discrete elements like strings of arbitrarily long length. Uncountable things are things like the space of real values $\mathbb{R}$.

It's useful to think of two extrema of entropy. The first is, for a finite (not countably infinite) space $\mathcal{X}$, what's the maximum possible entropy of a probability distribution over that space? Well, we're maximally uncertain about the value of a random variable if it could be any of them with equal probability. Let $p_{\text{uniform}}(x) = \frac{1}{|\mathcal{X}|}$. This is called the uniform distribution over $\mathcal{X}$. The entropy of this distribution is

$$H(p_{\text{uniform}}) = -\sum_{x \in \mathcal{X}} p_{\text{uniform}}(x) \log p_{\text{uniform}}(x) \tag{33}$$

$$= -|\mathcal{X}| \left(\frac{1}{|\mathcal{X}|} \log \frac{1}{|\mathcal{X}|} \right) \tag{34}$$

$$= -\log \frac{1}{|\mathcal{X}|} = \log |\mathcal{X}| \tag{35}$$

So, the maximum possible entropy grows with the log of the size of the finite space.

What's the minimum entropy a distribution can have and how do we achieve that?

Comparing distributions

One way of measuring how similar two distributions are is the Kullback-Leibler divergence, which looks a lot like entropy. Let p and q be distributions over the same countable space $\mathcal{X}$. Then the KL-divergence is:

$$D_{\text{KL}}(p \| q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \tag{36}$$

The KL-divergence is a directed quantity. Intuitively, $D_{\text{KL}}(p \| q)$ views p as a reference distribution, or true distribution. (In the sense that we're weighting each term in x by $p(x)$, so the "importance" or likelihood of each element in computing the divergence is weighted by p. The term in the fraction is a ratio of probabilities, $\frac{p(x)}{q(x)}$, so you should think of the KL-divergence as caring about probability ratios, not, say, absolute differences of probabilities. If two probabilities are really small, like 10^{-5} and 10^{-6}, this can still matter a lot. Finally, consider that if we fix $p(x)$ and slowly push $q(x)$ to zero, the term $\frac{p(x)}{q(x)}$ zooms off to infinity, so if you want low KL, you really don't want there to be any element x such that $p(x) > 0$ and $q(x) = 0$. Contrastively, if $q(x)$ is large but $p(x) = 0$, it doesn't matter, as the contribution of this element to the divergence is zero.

If KL-divergence cares about probability ratios, the **total variation distance** cares about absolute differences. If you recall the L_1 norm from vectors, this is basically that with a factor of $\frac{1}{2}$ multiplied in:

$$d_{\text{TV}}(p, q) = \frac{1}{2} \sum_{x \in \mathcal{X}} |p(x) - q(x)| \tag{37}$$

One way to think of the total variation distance is, if I were to sample from p and I were to sample from q, what fraction of the time would those samples be different elements of $\mathcal{X}$?

Joint and conditional distributions

Joint probability distributions express the uncertainty we have over a set of random variables. A probability distribution p over a space $\mathcal{X} \times \mathcal{Y}$ of tuples of elements, one from space $\mathcal{X}$ and one from space $\mathcal{Y}$, can be seen as a function that maps pairs (x, y) to scalar probabilities $p(x, y)$, where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.

Conditional probability distributions express the uncertainty we have over the value of one random variable given that we're told the value of another. Taking a joint distribution $p(x, y)$, the conditional distribution $p(y \mid x)$ states the probability of y given an observed value of x. One can think of this as taking the joint probability and fixing x, so that we have now a single-variable distribution over $\mathcal{Y}$.

A related concept is **marginalization**, in which we take a joint distribution $p(x, y)$, and derive from it a distribution $p(x)$ wherein we've "marginalized out" the variable y by considering all values y could take, considering the probabilities thereof, and summing over these probabilities to get the probabilities of just x without dependence on y:

$$p(x) = \sum_{y \in \mathcal{Y}} p(x, y) \tag{38}$$

Chain rule of probability

Consider a large set of spaces $\{\mathcal{X}_1, \ldots, \mathcal{X}_m\}$, and the goal of representing a probability distribution over that space. On the one hand, such a distribution is a function that takes in a tuple like $(x_1, \ldots, x_m)$ and produces a scalar. But it's often useful to express this joint distribution as a product of univariate conditional distributions. As an example, if I have a joint distribution over a variable representing whether I pass a test and a variable representing whether I study, it might be useful to model (1) whether I study, and then separately (2) whether I pass conditioned on knowing I did/didn't study.

The following identity, which equates the joint probability to a product of conditionals, is called the chain rule of probability:

$$p(x_1, \ldots, x_m) = \prod_{i=1}^{m} p(x_i \mid x_{<i}) \tag{39}$$

Matrix calculus

In some sense, all of the learning done by the systems in this course is made possible by neat software that performs efficient autodifferentiation of the functions that we specify. So, on a day to day basis we don't spend a ton of time computing gradients. However, sometimes we do—often when defining a new loss function or regularization—and regardless it's important to know for debugging your network training. (My network isn't training! Oh maybe the gradient is zero through the blahblah function...)

Vector gradients

You're likely familiar with differential calculus:

$$\frac{\partial}{\partial x} x^2 y = 2xy \tag{40}$$

For what we need of matrix calculus, we're basically doing the same thing, but we get a vector-valued gradient. Consider two vectors, $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$. Let $f(\mathbf{u}, \mathbf{v}) = \mathbf{u}^\top \mathbf{v}$. Then the partial derivative of f with respect to u we write as

$$\nabla_{\mathbf{u}} f = \nabla_{\mathbf{u}} \mathbf{u}^\top \mathbf{v} \tag{41}$$

$$= \mathbf{v} \tag{42}$$

How can you tell? Well, you can take a single-variable derivative for each u_i:

$$\begin{aligned}
\nabla_{u_i} f &= \nabla_{u_i} \mathbf{u}^\top \mathbf{v} && && (43)\\
&= \nabla_{u_i} \sum_{j=1}^{d} u_i v_i && \text{re-writing dot product as sum} && (44)\\
&= \sum_{j=1}^{d} \nabla_{u_i} u_j v_j && \text{derivative passes through sum} && (45)\\
&= \nabla_{u_i} u_i v_i + \sum_{j \neq i} \nabla_{u_j} u_j v_j && \text{split into term with } u_i \text{ and others} && (46)\\
&= \nabla_{u_i} u_i v_i + \sum_{j \neq i} 0 && \text{terms without } u_i \text{ are constant w.r.t. } u_i && (47)\\
&= v_i && \text{derivative of a linear function} && (48)
\end{aligned}$$

Computing this single-term gradient for every i gives you a vector of gradients $\nabla_{\mathbf{u}} f = [v_1, \ldots, v_d] = \mathbf{v}$.

Matrix-vector gradients

Let $W \in \mathbb{R}^{m \times d}$ and $\mathbf{v} \in \mathbb{R}^d$. Let $f(W, v) = W\mathbf{v}$. What's the gradient with respect to W? Well, it's not defined, because f isn't a scalar function. Even though the form of the gradient ∇_W will be of shape $\mathbb{R}^{m \times d}$, we can't take the gradient of a vector function. This is sometimes a gotcha; here are some quantities you might be interested in computing instead.

Let $f(W, v) = (W\mathbf{v})_i$. This function returns the i^{th} dimension of the vector result of $W\mathbf{v}$. This is a scalar function! The result is:

$$\begin{aligned}
\nabla_W f(W, v) &= \nabla_W (W\mathbf{v})_i && && (49)\\
&= \nabla_W W_{i,:}^\top \mathbf{v} && && (50)\\
&= [\cdots; \mathbf{0}; \mathbf{v}; \mathbf{0}; \cdots] \in \mathbb{R}^{m \times d} && \mathbf{v} \text{ at index } i && (51)\\
& && && (52)
\end{aligned}$$

This notation might be a bit confusing. Writing out $(W\mathbf{v})_i$ as $W_{i,:}^\top \mathbf{v}$, we're noting that only the row $W_{i,:}^\top \mathbf{v}$ contributes to the result at index i. This gives us a vector-vector gradient; the gradient w.r.t. $W_{i,:}^\top \mathbf{v}$ of v_i is $\mathbf{v}$. So, the gradient of all of W is a bunch of zeros everywhere except at index i out of m.

Chain rule of differential calculus, plus matrix conventions

Also recall taking derivatives through nonlinearities. Let $W \in \mathbb{R}^{m \times d}$, let $\mathbf{u} \in \mathbb{R}^m$, and let $\mathbf{v} \in \mathbb{R}^d$. Let

$$f(W, \mathbf{u}, \mathbf{v}) = \mathbf{u}^\top \exp(W\mathbf{v}) \tag{53}$$

Let's compute the gradient with respect to $\mathbf{v}$:

$$\nabla_{\mathbf{v}} f(W, \mathbf{u}, \mathbf{v}) = \nabla_{\mathbf{v}} \mathbf{u}^\top \exp(W\mathbf{v}) \tag{54}$$

$$= \nabla_{\mathbf{v}} \sum_{i=1}^{m} u_i \exp(W_{i,:}^\top \mathbf{v}) \qquad \text{dot product of } \mathbf{u} \text{ and } \exp(\cdot) \text{ as a sum} \tag{55}$$

$$= \sum_{i=1}^{m} u_i \nabla_{\mathbf{v}} \exp(W_{i,:}^\top \mathbf{v}) \qquad \text{gradient passes through sum and constant multiple} \tag{56}$$

$$= \sum_{i=1}^{m} u_i \exp(W_{i,:}^\top \mathbf{v}) \nabla_{\mathbf{v}} W_{i,:}^\top \mathbf{v} \qquad \text{chain rule} \tag{57}$$

$$= \sum_{i=1}^{m} u_i \exp(W_{i,:}^\top \mathbf{v}) W_{i,:} \qquad \text{vector-vector gradient} \tag{58}$$

You're welcome to leave the gradient like this. u_i is a scalar, $\exp(W_{i,:}^\top \mathbf{v})$ is a scalar, and we sum over all $W_{i,:}$ multiplied by these scalars to get the gradient for $\mathbf{v}$. But we can also write this sum in a matrix form by writing out our scalars $\exp(W_{i,:}^\top \mathbf{v})$ as a *diagonal* matrix with the scalar values around the diagonal, as such:

The m vectors of shape $\mathbb{R}^d$ of this matrix are being weighted and summed

$$\sum_{i=1}^{m} u_i \exp(W_{i,:}^\top \mathbf{v}) W_{i,:} = (\mathbf{u}^\top \operatorname{diag}(\exp(W\mathbf{v}))) W \tag{59}$$

Diagonal matrix, $\mathbb{R}^{m \times m}$, where $\operatorname{diag}(\exp(W\mathbf{v}))_{ii} = W_{i,:}^\top \mathbf{v}$

We've highlighted here the correspondence between terms on the left- and right-hand-side. The use of the diagonal matrix is a convention for concisely writing out the gradient through an elementwise nonlinearity (like the exponential, here.) It makes the matrix shapes work out. Another equivalent way to write this (with identical highlighting) is

The $\mathbb{R}^d$ vectors of this matrix are being weighted and summed

$$\sum_{i=1}^{m} u_i \exp(W_{i,:}^\top \mathbf{v}) W_{i,:} = (\mathbf{u} \odot \exp(W\mathbf{v}))^\top W \tag{60}$$

This is a vector in $\mathbb{R}^m$, like $\mathbf{u}$.

In this notation, we take the vector $\mathbf{u}$ and take its *elementwise product* (aka *hadamard product*) with the vector $\exp(W\mathbf{v})$. The elementwise product is

$$(\mathbf{u} \odot \exp(W\mathbf{v}))_i = u_i \exp(W\mathbf{v})_i \tag{61}$$

where u_i and $\exp(W\mathbf{v})_i$ are scalars. To see the equivalence between this matrix form and the previous one, note that the result of $\mathbf{u}^\top(\operatorname{diag}(\exp(W\mathbf{v})))$ is an $\mathbb{R}^m$ vector with the same values.