Mathematics behind Supervised Machine Learning Algorithms

Narsimha Chilkuri

University of Waterloo

Contents

There are a few reasons as to why I am writing these notes:

1. Writing forces me to think at a much higher resolution and this often brings me into contact with a number of misunderstood concepts.
2. Writing things down, especially electronically, makes it available forever for future reference.
3. Apparently, writing frees up mental space and this freed-up memory can be used to learn newer things.

Acknowledgements: These notes are far from original; they are based on a number of texts and online-courses. A few of them, in no particular order, are: Introduction to Statistical Thinking (by G. James, D. Witten etc), Deep Learning (by I. Goodfellow, Y. bengio etc), Learning from Data (course by Y.S Abu-Mostafa) and CS229 (course by Andrew Ng).

Introduction

As of now, there are three major paradigms in machine learning. They are supervised learning, unsupervised learning and reinforcement learning. I find it hard to say anything here that applies to all three types of learning, other than the fact that none of them make use of explicit programming to solve problems but instead rely on data, whether labelled (or instructive) (in-case of supervised), evaluative (in-case of reinforcement) or unlabelled (in-case of unsupervised), to solve problems of various kinds. Let us discuss each of these briefly down below.

Supervised Learning: Consider the following two conditions:

1. We have access to a dataset, D, consisting of pairs $(x,\, y)$:

$$\{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\},$$

where $x \in X \subset \mathbb{R}^d$ and $y \in Y \subset \mathbb{R}^l$.

2. We have reasons to believe that there exists a pattern to this data i.e., there is function f such that

$$\begin{aligned} Y &= f(X) + noise, \\ &= f(X) + \epsilon, \text{ where we assume } \epsilon \sim N(0, \sigma^2). \end{aligned}$$

Given the above two conditions, the goal of supervised learning is to obtain an *estimate* of this function f using nothing but the dataset D!

Unsupervised Learning:

Reinforcement Learning: The main idea of reinforcement learning is to learn by interacting with the environment in order to achieve a goal. More specifically, we deal with a Markov Decision Process (MDP) where the transition probabilities and rewards are unknown before-hand and the goal is to learn a policy.

1 K-Nearest Neighbours

If I were thinking about the data-driven approach to solving problems all by myself, I believe I would have discovered the K-Nearest Neighbours (KNN) approach first. This approach is not only intuitive, but also uses elementary mathematics. The crux of this approach can be stated as follows: a point has similar properties as the points lying in its proximity. Mathematically, we can write the previous statement as:

$$f(\vec{x}_*) = \frac{1}{K} \sum_{i=1}^{K} f(\vec{x}_i), \tag{1.1}$$

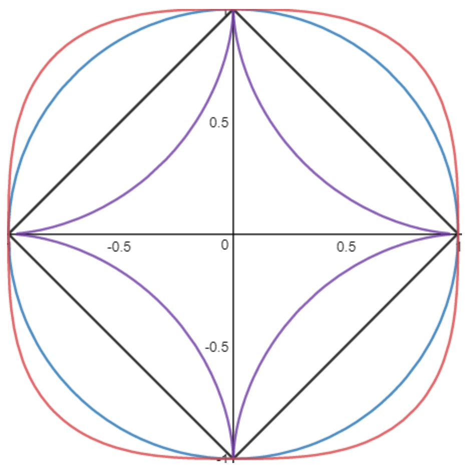

Figure 1.

i.e., we predict the output of a point $\vec{x}_*$ by averaging the outputs of K nearest points.

I have written down eq.(1.1), but I have made no mention of what K should be. Is there a formula relating K to the number of training examples, dimension of the problem, distribution of the training and testing data etc.? Unfortunately, as is presently the case with most of the algorithms in machine learning, there is no standard answer for values of parameters such as K that work well in all scenarios. It has to be figured out by trial and error.

Another thing that I have left out is the meaning of "near" (in nearest K neighbours) or the choice of the distance function. We use a generalized form of distance called the Minkowski metric defined as:

$$Distance(\vec{x}, \vec{y}; p) = \left(\sum_i |x_i - y_i|^p \right)^{\frac{1}{p}},$$

where x_i and y_i are components of the vectors $\vec{x}$ and $\vec{y}$. In figure 1 we see all the points that are equidistant from the origin for various values of p. Note that we obtain the familiar Euclidean distance if we plug $p = 2$ in the above equation. Before we discuss more about this, let us talk a bit about how our intuition breaks down in "high" dimensions.

2 Linear Regression

Linear regression is a relatively old technique, first published by Legendre in 1805 and then by Gauss in 1809. They developed the method to make sense of astronomical data.

Consider a dataset $\{\vec{x}_i, y_i\}$, where $\vec{x}_i$ is an n-dimensional input vector of real values and y_i is the corresponding real valued output. Denoting the predicted output for a given input $\vec{x}_i$ by $\hat{y}_i$, we can define the residual squares sum (RSS) term as:

$$RSS = \sum_i (\hat{y}_i - y_i)^2.$$

Here, we make an assumption which explains the term 'linear' in linear-regression. We assume that the *true* function f that relates Y and X is a linear function of X:

$$\begin{aligned} &Y = f(X) + \epsilon, \\ &Y = \vec{w} \cdot \vec{X} + b + \epsilon, \text{ this is the linear assumption,} \end{aligned}$$

and our job is to estimate the function f, or, in other words, estimate the parameters $\vec{w}$ and b. We denote the estimates using a hat, $\hat{\vec{w}}$ and $\hat{b}$. Once we have these estimates of parameters, we can compute predictions as follows:

$$\hat{y}_i = \hat{\vec{w}} \cdot \vec{x}_i + b.$$

Plugging in the above assumption into eq (4), we obtain the following:

$$E = \sum_i ((\hat{\vec{w}} \cdot \vec{x}_i + b) - y_i)^2.$$

The above equation can be written in a slightly different form, where the intercept term (also called bias) b is absorbed into the coefficients vector (also called weight vector). This is possible if we define another vector $\hat{\vec{w}}' = [\hat{b}, \hat{w}_1, \hat{w}_2, ...]$ and $\vec{x}_i' = [1, x_{1i}, x_{2i}, ...]$. We can now write:

$$RSS = \sum_i (\hat{\vec{w}}' \cdot \vec{x}_i' - \vec{y}_i)^2.$$

It is a lot cleaner to solve the above minimization problem if we write the above equation in-terms of matrices and vectors. When this is done, we will be left with the following equation:

$$RSS = ||X\hat{\vec{W}} - Y||^2,$$

where the matrix X is made up by stacking the input vectors $\vec{x}_i'$ in rows, $\vec{Y}$ is the vector containing all the output values y_i and $\hat{\vec{W}} = \hat{\vec{w}}'$.

Minimizing the Error

We now minimize the error term with respect to the weights $\hat{\vec{W}}$. Differentiating the error term with respect to $\hat{\vec{W}}$ and setting it to zero, we obtain:

$$\begin{aligned} 2X^T(X\hat{\vec{W}} - Y) = 0, \\ \rightarrow X^T X\hat{\vec{W}} - X^T Y = 0. \end{aligned}$$

If X has more rows than columns, then it is very likely that the columns are independent and this implies that X^TX is also invertible. Assuming X has independent columns, we can multiply the above equation by the inverse of X^TX to end up with:

$$\hat{\vec{W}} = (X^TX)^{-1}X^TY.$$

We thus have a nice closed form solution for the weigh vector. This is a sight to be cherished as something like this is hard to come by in machine-learning.

Statistical Inference for Linear Regression

One of the main advantages with linear regression is that it lends itself very well to statical analysis, thus permitting us to construct confidence-intervals and to perform hypothesis testing. Before we dive into this, let us look at the problem of confidence intervals and such for a far simpler case.

From introductory statistics, consider the classic problem of estimating the mean height of a population; we do not know anything about the population, but we do have access to a sample of size N. Let us call the sample mean $\hat{\mu}$ and sample standard deviation $\hat{\sigma}$. Given this information, the tools of statistical-inference help us construct a confidence interval as:

$$\left[\hat{\mu} - t^*_{n-1}\frac{\hat{\sigma}}{\sqrt{n}}, \hat{\mu} + t^*_{n-1}\frac{\hat{\sigma}}{\sqrt{n}}\right].$$

In a similar fashion, we can construct confidence intervals for $\hat{\vec{W}}$ as shown below:

$$\left[\hat{w}_i - t^*_{n-p}SE(\hat{w}_i), \hat{\mu} + t^*_{n-p}SE(\hat{w}_i)\right].$$

It can be shown that the estimates come from a t_{n-p} distribution with mean and variance derived below:

$$\begin{aligned}
E[\hat{\vec{W}}] =& E[(X^TX)^{-1}X^TY] \\
=& E[(X^TX)^{-1}X^T(X\vec{W} + \epsilon)] \\
=& E[(X^TX)^{-1}X^TX\vec{W} + (X^TX)^{-1}X^T\epsilon)] \\
=& 0 \\
var(\vec{W}) =& var((X^TX)^{-1}X^TY) \\
=& var((X^TX)^{-1}X^T(X\vec{W} + \epsilon)) \\
=& var((X^TX)^{-1}X^TX\vec{W} + (X^TX)^{-1}X^T\epsilon))
\end{aligned}$$

Using $var(A\vec{B}+\vec{a}) = A\ var(\vec{B})\ A^T$ (where $\vec{a}$ is a constant), the above equation can be written as:

$$\begin{aligned}
var(\vec{W}) =& (X^TX)^{-1}X^T\ var(\epsilon)\ ((X^TX)^{-1}X^T)^T \\
=& (X^TX)^{-1}X^T\sigma^2\ \mathbf{I}\ ((X^TX)^{-1}X^T)^T \\
=& (X^TX)^{-1}X^T\sigma^2\ \mathbf{I}\ X(X^TX)^{-1} \\
\rightarrow var(\vec{W}) =& \sigma^2(X^TX)^{-1}
\end{aligned}$$

Thus we obtain the co-variance matrix for the estimated parameters $\vec{W}$.

3 Modifications to Linear Regression

Taylor and Fourier Bases

In-order to deal with non-linear data, we will have to relax the assumption that we made in the earlier section i.e., instead of assuming that the predicted output is a linear function of the input, we consider the output to be any general function of the input. We can demonstrate this in one-dimension as follows:

$$\begin{aligned}
linear : \hat{y}_i &= wx_i + b \\
general : \hat{y}_i &= b + w_1x_i + w_2x_i^2 + w_3x_i^3 + ... \\
general : \hat{y}_i &= b + w_1\cos(x_i) + w_2\sin(x_i) + w_3\sin(2x_i) + w_4\cos(2x_i) + ...
\end{aligned}$$

We can recognize the above two general functions as the Taylor and Fourier expansions of a function. Let us pick the Fourier basis to fit non-linear data as the polynomial one is dealt with in many books. Although by the looks of it, the non-linear problem looks quite different from the linear one, we can convert this problem into pretty much the exact same one as before.

$$\begin{aligned}
x_i &\rightarrow [1, \cos(x_i), \sin(x_i), \sin(2x_i), \cos(2x_i), ...], \\
X &\rightarrow \begin{bmatrix} 1 & \cos(x_1) & \sin(x_1) & \sin(2x_1) & \cos(2x_1)... \\ 1 & \cos(x_2) & \sin(x_2) & \sin(2x_2) & \cos(2x_2)... \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}, \\
\vec{W} &\rightarrow [b, w_1, w_2, w_3, ...].
\end{aligned}$$

Using the above definitions, we can again compute the weight vector using the same equation as before:

$$\vec{W} = (X^TX)^{-1}X^TY.$$

4 Regularization for Linear Regression

Ridge Regularization

Compared to linear regression, here we minimize a slightly modified error term; to the residual-squared-sum, we add another term as shown below.

$$E_{ridge} = ||X\hat{\vec{W}} - \vec{Y}||^2 + \lambda||\hat{\vec{W}}||_2^2.$$

Differentiating with respect to $\hat{\vec{W}}$ and setting it to zero, we obtain:

$$\begin{aligned}
&2X^T(X\hat{\vec{W}} - \vec{Y}) + 2\lambda\hat{\vec{W}} = 0, \\
\rightarrow &X^TX\hat{\vec{W}} - X^T\vec{Y} + \lambda\hat{\vec{W}} = 0, \\
\rightarrow &\left(X^TX + \lambda\mathbf{I}\right)\hat{\vec{W}} - X^T\vec{Y} = 0, \\
\rightarrow &\hat{\vec{W}} = \left(X^TX + \lambda\mathbf{I}\right)^{-1}X^T\vec{Y}.
\end{aligned}$$

Lasso Regularization

$$E_{lasso} = ||X\hat{\vec{W}} - \vec{Y}||^2 + \lambda||\hat{\vec{W}}||_1.$$

For lasso, there is no closed form solution for the general X. We use coordinate descent to arrive at the parameters that minimize the loss function.

5 Logistic Regression

Consider a multi-class dataset $\{x^{(i)}, y^{(i)}\}$, where $i = 0, 1, 2, .., m$. x is some N-dimensional vector and, since we are dealing with a multi-class dataset, y is a K-dimensional vector with only one element set to 1 and the rest set to zero. The parameters W and b are a K by N dimensional matrix and a K dimensional vector respectively. Using these parameters, we predict the probability as given below:

$$\Pr(Y = y^{(i)}|X = x^{(i)}) = p(x^{(i)}) = \frac{1}{\sum_{r=1}^{K}\exp(\vec{w}^r \cdot x^{(i)} + b_r)}\begin{bmatrix}\exp(\vec{w}^1 \cdot x^{(i)} + b_1) \\ \exp(\vec{w}^2 \cdot x^{(i)} + b_2) \\ \vdots \\ \exp(\vec{w}^K \cdot x^{(i)} + b_K)\end{bmatrix}.$$

We can now represent the log-likelihood function as follows:

$$\mathcal{L} = \sum_{i=1}^{m}\sum_{j=1}^{K}\log(p_j(x^{(i)}))\,y_j^{(i)}.$$

Differentiating the loss with an arbitrary element of a vector $\vec{w}^m_n$, we get:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \vec{w}^r_s} &= \sum_{i=1}^{m} \frac{\partial}{\partial \vec{w}^r_s} \left(\log(p_r(x^{(i)}))\right) y^{(i)}_r, \\
&= \sum_{i=1}^{m} \frac{\partial}{\partial \vec{w}^r_s} \left(\log\left(\frac{\exp(\vec{w}^r \cdot x^{(i)} + b_r)}{\sum_r \exp(\vec{w}^r \cdot x^{(i)} + b_r)}\right)\right) y^{(i)}_r, \\
&= \sum_{i=1}^{m} \frac{\partial}{\partial \vec{w}^r_s} \left((\vec{w}^r \cdot x^{(i)} + b_r) - \log\left(\sum_r \exp(\vec{w}^r \cdot x^{(i)} + b_r)\right)\right) y^{(i)}_r, \\
&= \sum_{i=1}^{m} \left(x^{(i)}_s - \frac{\exp(\vec{w}^r \cdot x^{(i)} + b_r)\, x^{(i)}_s}{\sum_r \exp(\vec{w}^r \cdot x^{(i)} + b_r)}\right) y^{(i)}_r, \\
&= \sum_{i=1}^{m} x^{(i)}_s \left(1 - p_r(x^{(i)})\right) y^{(i)}_r.
\end{aligned}$$

Similarly, we have:

$$\frac{\partial \mathcal{L}}{\partial \vec{b}_r} = \sum_{i=1}^{m} \left(1 - p_r(x^{(i)})\right) y^{(i)}_r.$$

The above results can be used to carry out optimization by gradient descent.

6 Linear Discriminant Analysis

The Bayes' theorem sates that

$$\Pr(Y = k|X = x) = \frac{\Pr(Y = k)\Pr(X = x|Y = k)}{\sum_{i=1}^{K} \Pr(Y = k)\Pr(X = x|Y = k)}.$$

Using the notation $\Pr(Y = k|X = x) = p_k(x)$, $\Pr(Y = k) = \pi_k$ and $\Pr(X = x|Y = k) = f_k(x)$, we can write the above theorem as:

$$p_k(x) = \frac{\pi_k f_k(x)}{\sum_{i=1}^{K} \pi_k f_k(x)}.$$

Linear Discriminant Analysis (LDA) makes use of the Bayes' theorem and a few assumptions on the probability density function, $f_k(x)$, in-order to classify points. The assumptions on the density function are: (1) it is gaussian with mean μ_k and variance Σ^2_k (2) $\Sigma_1 = \Sigma_2 = ... = \Sigma_K = \Sigma$.

For a given set of predictors x, LDA classification is done as follows:

$$\hat{y} = \text{argmax}_k\ p_k(x).$$

Since the denominator term in $p_k(x)$ is simply a scaling factor, we can instead do the classification as:

$$\hat{y} = \text{argmax}_k \ \pi_k f_k(x).$$

We simplify the above expression further by taking the log of the function that is being maximized.

$$\begin{aligned}
\log(\pi_k f_k(x)) &= \log(\pi_k) + log(f_k(x)),\\
&= \log(\pi_k) + \log\left(\frac{1}{(2\pi)^{\frac{p}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(x-\mu_k)^T\boldsymbol{\Sigma}^{-1}(x-\mu_k)\right)\right),\\
&= \log(\pi_k) + \log\left(\frac{1}{(2\pi)^{\frac{p}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}}\right) + \log\left(\exp\left(-\frac{1}{2}(x-\mu_k)^T\boldsymbol{\Sigma}^{-1}(x-\mu_k)\right)\right),\\
&\text{(ignoring the term common to all densities)}\\
&= \log(\pi_k) + \log\left(\exp\left(-\frac{1}{2}(x-\mu_k)^T\boldsymbol{\Sigma}^{-1}(x-\mu_k)\right)\right),\\
&= \log(\pi_k) - \frac{1}{2}(x-\mu_k)^T\boldsymbol{\Sigma}^{-1}(x-\mu_k),\\
&= \log(\pi_k) - \frac{1}{2}\left[x^T\boldsymbol{\Sigma}^{-1}x - x^T\boldsymbol{\Sigma}^{-1}\mu_k - \mu_k^T\boldsymbol{\Sigma}^{-1}x + \mu_k^T\boldsymbol{\Sigma}^{-1}\mu_k\right],\\
&\text{(again ignoring the term common to all densities)}\\
&= \log(\pi_k) - \frac{1}{2}\left[-x^T\boldsymbol{\Sigma}^{-1}\mu_k - \mu_k^T\boldsymbol{\Sigma}^{-1}x + \mu_k^T\boldsymbol{\Sigma}^{-1}\mu_k\right],\\
&\text{(using } x^T\boldsymbol{\Sigma}^{-1}\mu_k = \mu_k^T\boldsymbol{\Sigma}^{-1}x\text{)}\\
&= \log(\pi_k) + x^T\boldsymbol{\Sigma}^{-1}\mu_k - \frac{1}{2}\mu_k^T\boldsymbol{\Sigma}^{-1}\mu_k.
\end{aligned}$$

Hence, we can write:

$$\hat{y} = \text{argmax}_k \ \log(\pi_k) + x^T\boldsymbol{\Sigma}^{-1}\mu_k - \frac{1}{2}\mu_k^T\boldsymbol{\Sigma}^{-1}\mu_k.$$

7 Quadratic Discriminant Analysis

Quadratic discriminant analysis is exactly like LDA but without the second assumption that the density distributions have equal variances. Therefore, without the assumption that $\Sigma_1 = \Sigma_2 = ... = \Sigma_K = \Sigma$, we can no longer ignore the terms we ignored in the above derivation. Thus we classify using the following equation:

$$\begin{aligned}
\hat{y} &= \text{argmax}_k \ \log(\pi_k) + x^T\boldsymbol{\Sigma_k}^{-1}\mu_k - \frac{1}{2}\mu_k^T\boldsymbol{\Sigma_k}^{-1}\mu_k - \frac{1}{2}x^T\boldsymbol{\Sigma_k}^{-1}x + \log\left(\frac{1}{(2\pi)^{\frac{p}{2}}|\boldsymbol{\Sigma_k}|^{\frac{1}{2}}}\right),\\
&= \text{argmax}_k \ \log(\pi_k) + x^T\boldsymbol{\Sigma_k}^{-1}\mu_k - \frac{1}{2}\mu_k^T\boldsymbol{\Sigma_k}^{-1}\mu_k - \frac{1}{2}x^T\boldsymbol{\Sigma_k}^{-1}x - \frac{1}{2}\log(|\boldsymbol{\Sigma_k}|).
\end{aligned}$$

8 Support Vector Machine

The building block of support vector machine (SVM) is what is known as the maximal margin classifier (MMC). MMC is a simple classifier which, given that the data is linearly separable, constructs a *separating* hyperplane such that the smallest perpendicular distance from the training observations to the hyperplane is maximized. The smallest perpendicular distance from the training observations to the separating hyperplane is called the margin, and hence the name maximal margin classifier.

In order to deal with scenarios where the data is not exactly linearly separable, the MMC is modified to have soft margins i.e.,the margin is still maximized but we do not require the hyperplane to be exactly separating. This modified version of MMC is known as the soft margin classifier.

Finally, we modify the soft margin classifier to deal with non-linear data classification in a very clever way which allows us to go to very high dimensions – even ∞ dimensions! – without really paying the computational price. This modified classifier is known as the support vector machine.

The optimization problem for SVM is given below:

$$\begin{aligned}
&\min_{\gamma,w,b}\ \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\epsilon_i \\
&\text{subject to } y^{(i)}(w^T x^{(i)} + b) \geq 1 - \epsilon_i,\ \ i = 0,1,2,...,m, \\
&\epsilon_i \geq 0,\ \ i = 0,1,2,...,m.
\end{aligned}$$

As you might have guessed, there is no exact solution for the above problem. Optimization is usually done using the SMO algorithm which makes use of coordinate descent.

9 Back-propagation for binary Neural Networks

Let us consider a general neural network. No matter what the structure of the network is, given we are dealing with binary classification, the last layer will contain only one node and the activation function for the last year will be the sigmoid function. Let us do the computation for derivatives of the parameters W and b in detail for the last layer of this general network i.e, $W^{[L]}, b^{[L]}$ and also $A^{[L-1]}$, where L denotes the final or output layer. One we have these, it is not too hard to see the jump to the general result.

$$\mathcal{L} = -\sum_{i=1}^{N} y^{(i)}\log(a^{[L](i)}) + (1 - y^{(i)})\log(1 - a^{[L](i)})$$

1 Differentiating with respect to $a^{[L](j)}$, we get:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial a^{[L](j)}} &= -\sum_{i=1}^{N} \frac{\partial}{\partial a^{[L](j)}} \left[y^{(i)} \log(a^{[L](i)}) + (1 - y^{(i)}) \log(1 - a^{[L](i)})\right] \\
&= -\frac{\partial}{\partial a^{[L](j)}} \left[y^{(j)} \log(a^{[L](j)}) + (1 - y^{(j)}) \log(1 - a^{[L](j)})\right] \\
&= -\left[\frac{y^{(j)}}{a^{[L](j)}} - \frac{(1 - y^{(j)})}{1 - a^{[L](j)}}\right]
\end{aligned}$$

Vectorized form of the above equation that lets us compute that above derivative for all j cab be written as follows (we assume we are dealing with numpy arrays):

$$\frac{\partial \mathcal{L}}{\partial A^{[L]}} = -\left[\frac{Y}{A^{[L]}} - \frac{(1 - Y)}{a^{[L](j)}}\right]$$

2 Now lets compute the derivative of loss with respect to z.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial z^{[L](j)}} &= \frac{\partial \mathcal{L}}{\partial a^{[L](j)}} \frac{\partial a^{[L](j)}}{\partial z^{[L](j)}} \\
&= \frac{\partial \mathcal{L}}{\partial a^{[L](j)}} \frac{\partial g(z^{[L](j)})}{\partial z^{[L](j)}}, \text{ where } g \text{ is the activation function}
\end{aligned}$$

In case of binary classification, $g(z) = \sigma(z)$, where σ is the sigmoid function. Therefore:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial z^{[L](j)}} &= \frac{\partial \mathcal{L}}{\partial a^{[L](j)}} \frac{\partial \sigma(z^{[L](j)})}{\partial z^{[L](j)}} \\
&= \frac{\partial \mathcal{L}}{\partial a^{[L](j)}} \frac{\partial}{\partial z^{[L](j)}} \left[\frac{1}{1 + \exp(-z^{[L](j)})}\right] \\
&= \frac{\partial \mathcal{L}}{\partial a^{[L](j)}} \left(\frac{1}{1 + \exp(-z^{[L](j)})}\right) \left(1 - \frac{1}{1 + \exp(-z^{[L](j)})}\right) \\
&= \frac{\partial \mathcal{L}}{\partial a^{[L](j)}} a^{[L](j)}(1 - a^{[L](j)})
\end{aligned}$$

Again, we can write the above equation in vectorized form as:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial Z^{[L]}} &= \frac{\partial \mathcal{L}}{\partial A^{[L]}} * A^{[L]}(1 - A^{[L]}), \\
&= -\left[\frac{Y}{A^{[L]}} - \frac{(1 - Y)}{a^{[L](j)}}\right] * A^{[L]}(1 - A^{[L]}),
\end{aligned}$$

where '$*$' indicates element-wise multiplication.

3

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial W_j^{[L]}} &= \sum_{i=1}^{N} \frac{\partial \mathcal{L}}{\partial a^{[L](i)}} \frac{\partial a^{[L](i)}}{\partial z^{[L](i)}} \frac{\partial z^{[L](i)}}{\partial W_j^{[L]}} \\ &= \sum_{i=1}^{N} \frac{\partial \mathcal{L}}{\partial a^{[L](i)}} \frac{\partial a^{[L](i)}}{\partial z^{[L](i)}} \frac{\partial}{\partial W_j^{[L]}} \left(\sum_{l=1}^{n_{L-1}} a_l^{[L-1](j)} W_l^{[L]} + b_j^{[L]} \right) \\ &= \sum_{i=1}^{N} \frac{\partial \mathcal{L}}{\partial a^{[L](i)}} \frac{\partial a^{[L](i)}}{\partial z^{[L](i)}} a_j^{[L-1](i)}\end{aligned}$$

Therefore, the vectorized form is:

$$\frac{\partial \mathcal{L}}{\partial W^{[L]}} = \frac{\partial \mathcal{L}}{\partial Z^{[L]}} \cdot A^{[L-1]^T}.$$

4

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial b_j^{[L]}} &= \sum_{i=1}^{N} \frac{\partial \mathcal{L}}{\partial a^{[L](i)}} \frac{\partial a^{[L](i)}}{\partial z^{[L](i)}} \frac{\partial z^{[L](i)}}{\partial b_j^{[L]}} \\ &= \sum_{i=1}^{N} \frac{\partial \mathcal{L}}{\partial a^{[L](i)}} \frac{\partial a^{[L](i)}}{\partial z^{[L](i)}} \frac{\partial}{\partial b_j^{[L]}} \left(\sum_{l=1}^{n_{L-1}} a_l^{[L-1](j)} W_l^{[L]} + b_j^{[L]} \right) \\ &= \sum_{i=1}^{N} \frac{\partial \mathcal{L}}{\partial a^{[L](i)}} \frac{\partial a^{[L](i)}}{\partial z^{[L](i)}} (1)\end{aligned}$$

Therefore, the vectorized form is:

$$\frac{\partial \mathcal{L}}{\partial W^{[L]}} = \frac{\partial \mathcal{L}}{\partial Z^{[L]}} \cdot \mathbf{1},$$

where $\mathbf{1}$ is a row vector of ones of dimension Nx1.

5

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial a_j^{[L-1]}} &= \frac{\partial \mathcal{L}}{\partial a^{[L](i)}} \frac{\partial a^{[L](i)}}{\partial z^{[L](i)}} \frac{\partial z^{[L](i)}}{\partial a_j^{[L-1]}} \\ &= \frac{\partial \mathcal{L}}{\partial a^{[L](i)}} \frac{\partial a^{[L](i)}}{\partial z^{[L](i)}} \frac{\partial}{\partial a_j^{[L-1]}} \left(\sum_{l=1}^{n_{L-1}} a_l^{[L-1](j)} W_l^{[L]} + b_j^{[L]} \right) \\ &= \frac{\partial \mathcal{L}}{\partial a^{[L](i)}} \frac{\partial a^{[L](i)}}{\partial z^{[L](i)}} W_j^{[L]}\end{aligned}$$

Therefore, the vectorized form is:

$$\frac{\partial \mathcal{L}}{\partial A^{[L-1]}} = \frac{\partial \mathcal{L}}{\partial Z^{[L]}} * W^{[L]}.$$

Equations we derived

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial A^{[L]}} &= -\left[\frac{Y}{A^{[L]}} - \frac{(1-Y)}{A^{[L]}}\right] \\
\frac{\partial \mathcal{L}}{\partial Z^{[L]}} &= \frac{\partial \mathcal{L}}{\partial A^{[L]}} * A^{[L]}(1 - A^{[L]}) \\
\frac{\partial \mathcal{L}}{\partial W^{[L]}} &= \frac{\partial \mathcal{L}}{\partial Z^{[L]}} \cdot A^{[L-1]^T} \\
\frac{\partial \mathcal{L}}{\partial W^{[L]}} &= \frac{\partial \mathcal{L}}{\partial Z^{[L]}} \cdot \mathbf{1} \\
\frac{\partial \mathcal{L}}{\partial A^{[L-1]}} &= \frac{\partial \mathcal{L}}{\partial Z^{[L]}} * W^{[L]}
\end{aligned}
$$

General Equations

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial A^{[L]}} &= -\left[\frac{Y}{A^{[L]}} - \frac{(1-Y)}{A^{[L]}}\right] \\
\frac{\partial \mathcal{L}}{\partial Z^{[l]}} &= \frac{\partial \mathcal{L}}{\partial A^{[l]}} * g^{'}(Z^{[l]}) \\
\frac{\partial \mathcal{L}}{\partial W^{[l]}} &= \frac{\partial \mathcal{L}}{\partial Z^{[l]}} \cdot A^{[l-1]^T} \\
\frac{\partial \mathcal{L}}{\partial W^{[l]}} &= \frac{\partial \mathcal{L}}{\partial Z^{[l]}} \cdot \mathbf{1} \\
\frac{\partial \mathcal{L}}{\partial A^{[l-1]}} &= W^{[l]^T} \cdot \frac{\partial \mathcal{L}}{\partial Z^{[l]}}
\end{aligned}
$$