Hyperparameters

Hyperparameters are variables that control different aspects of training. Three common hyperparameters are:

- Learning rate
- Batch size
- Epochs

In contrast, parameters are the variables, like the weights and bias, that are part of the model itself.

Learning rate is a floating-point number you set that influences how quickly the model converges. If the learning rate is too low, the model can take a long time to converge. However, if the learning rate is too high, the model never converges, but instead bounces around the weights and bias that minimize the loss. The goal is to pick a learning rate that's not too high nor too low so that the model converges quickly.

For example, if the gradient's magnitude is 2.5 and the learning rate is 0.01, then the model will change the parameter by 0.025.

The ideal learning rate helps the model to converge within a reasonable number of iterations. In Figure, the loss curve shows the model significantly improving during the first 20 iterations before beginning to converge:

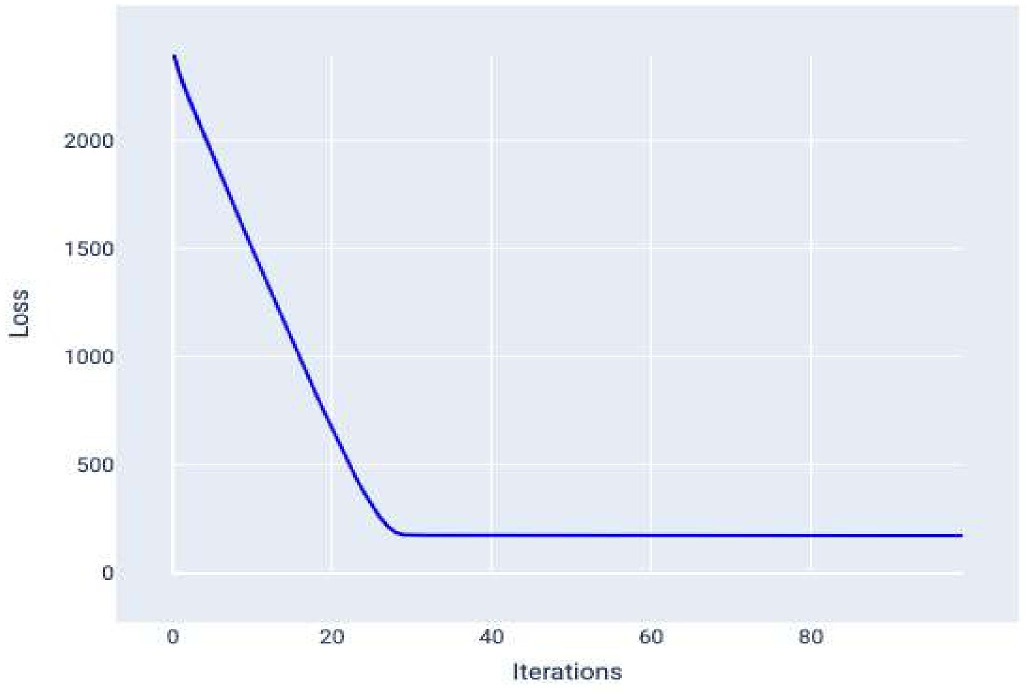


Figure 1. Loss graph showing a model trained with a learning rate that converges quickly.

In contrast, a learning rate that's too small can take too many iterations to converge. In Figure, the loss curve shows the model making only minor improvements after each iteration:

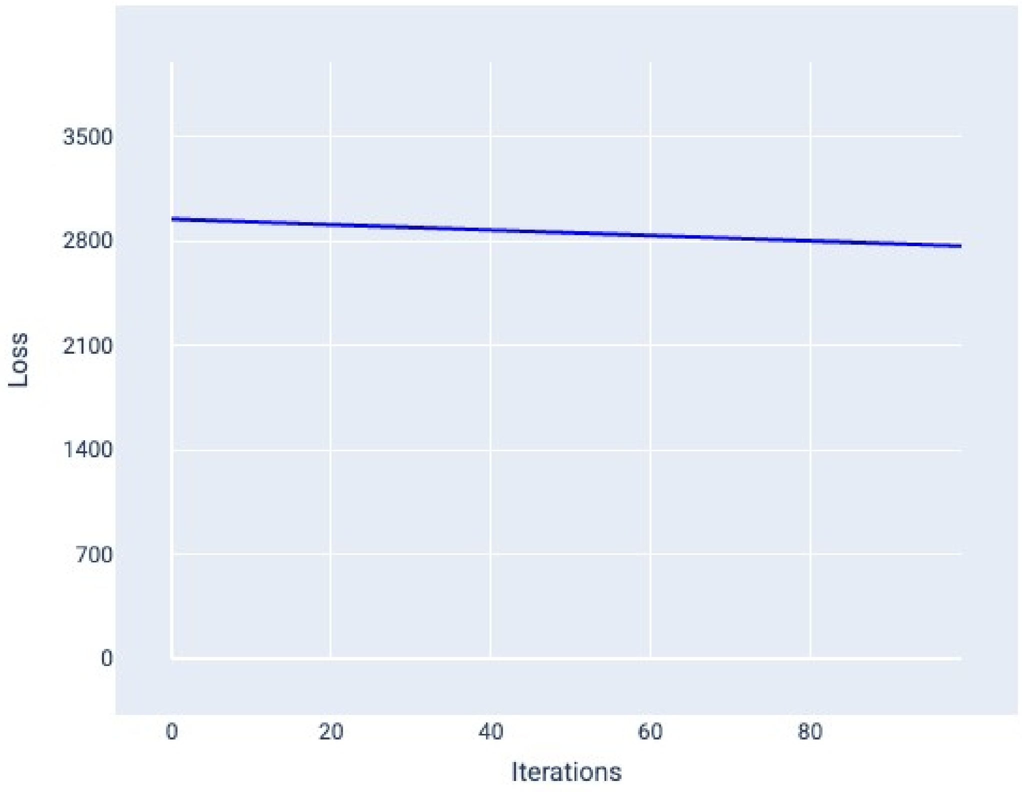


Figure 2. Loss graph showing a model trained with a small learning rate.

Batch size:

Batch size is a hyperparameter that refers to the number of times the model processes before updating its weights and bias.

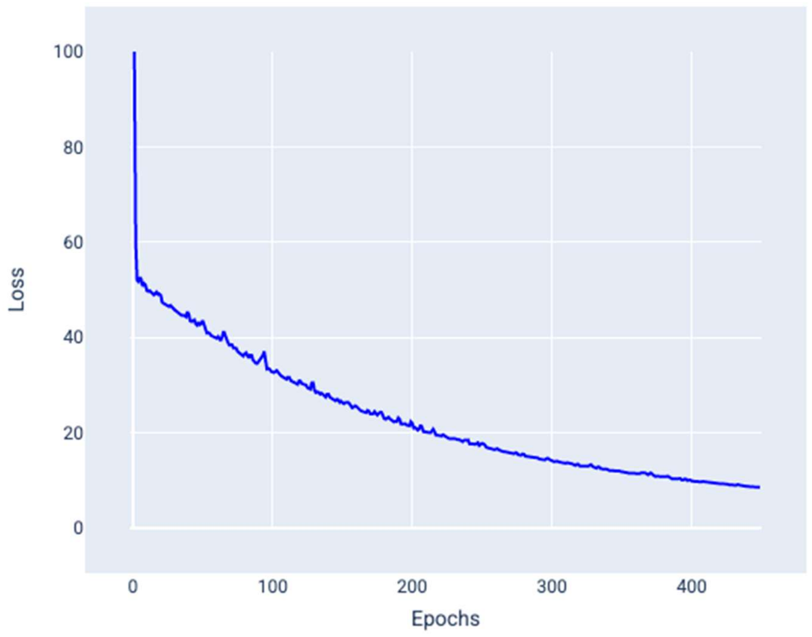
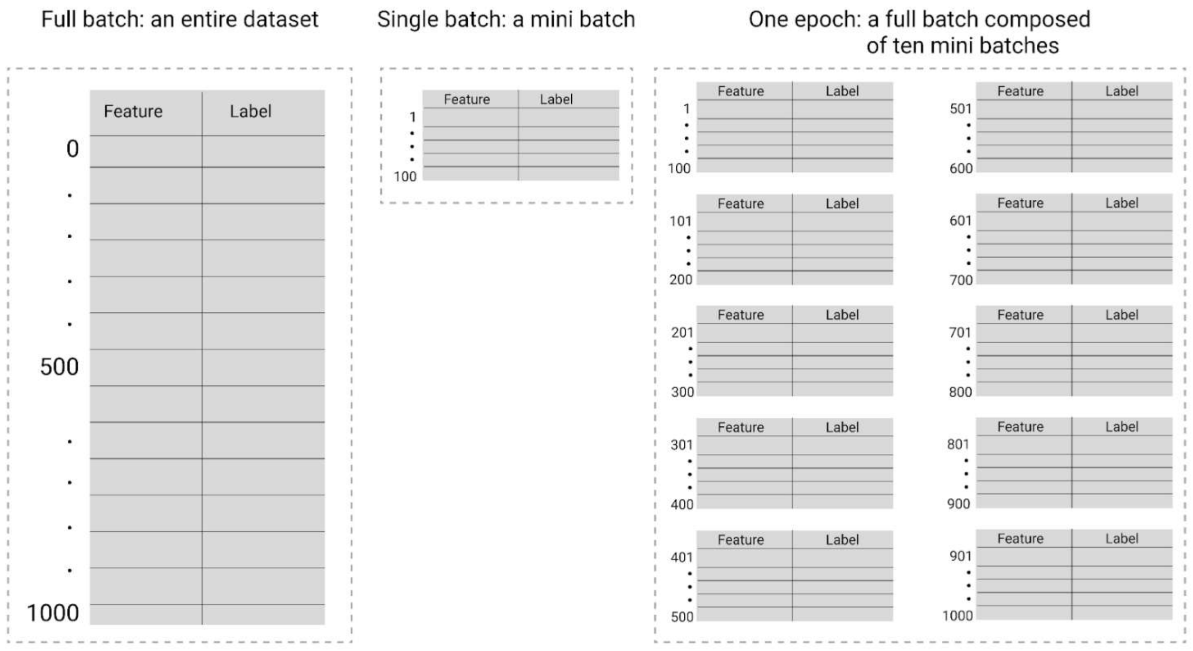


Figure 3. Model trained with mini-batch SGD.

Epochs

During training, an **epoch** means that the model has processed every example in the training set *once*. For example, given a training set with 1,000 examples and a mini-batch size of 100 examples, it will take the model 10 **iterations** to complete one epoch.



Full batch: an entire dataset
Single batch: a mini batch
One epoch: a full batch composed of ten mini batches
Feature
Label
0
500
1000
1
100
101
200
201
300
301
400
401
500
501
600
601
700
701
800
801
900
901
1000

Figure 3. Full batch versus mini batch

EVALUATION METRICS

TP (True Positive), FP (False Positive), and FN (False Negative) are key metrics from a confusion matrix used to evaluate a machine learning model's performance on a binary classification task. A True Positive (TP) is a correct positive prediction, a False Positive (FP) is a positive prediction where the actual value was negative (a Type I error), and a False Negative (FN) is a negative prediction where the actual value was positive (a Type II error).

Precision

Out of all the instances the model predicted as positive, what fraction were actually positive?

Recall

What it is: Out of all the actual positive instances, what fraction did the model correctly identify?

F1-Score

A single metric that combines precision and recall. It is the harmonic mean, which gives a higher weight to lower values. This means the F1-score will be low if either precision or recall is low.

Metric	Formula
True positive rate, recall	$\frac{\text{TP}}{\text{TP+FN}}$
False positive rate	$\frac{\text{FP}}{\text{FP+TN}}$
Precision	$\frac{\text{TP}}{\text{TP+FP}}$
Accuracy	$\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$
F-measure	$\frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$

Example 2.6 Let us map true positives, true negatives, false positives, and false negatives when $y = [1, 1, -1, 1, -1, -1, 1, 1, 1, 1]$ and $\hat{y} = [1, -1, 1, 1, -1, 1, 1, 1, 1, -1]$. Further, based on these counts, we can produce a confusion matrix.

In Table 2.9, we enlist the type of correct/incorrect information captured by the ith index. We can see that TP occurs when $y_i = \hat{y}_i = 1$ and TN at $y_i = \hat{y}_i = -1$. Meanwhile, at indices 2 and 10, we observe the case of $y_i = 1$ but $\hat{y}_i = -1$, causing false negatives. Finally, at indices 3 and 6, we note $y_i = -1$ but $\hat{y}_i = 1$, leading to false positives.

Now, mapping the type count in Table 2.9, we can construct the confusion matrix for the four cases as accounted in Table 2.10.

TABLE 2.9 Mapping true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) for expected labels $y = [1, 1, -1, 1, -1, -1, 1, 1, 1, 1]$ and predicted labels $\hat{y} = [1, -1, 1, 1, -1, 1, 1, 1, 1, -1]$.

Index	1	2	3	4	5	6	7	8	9	10
Expected y	1	1	−1	1	−1	−1	1	1	1	1
Predicted $\hat{y}$	1	−1	1	1	−1	1	1	1	1	−1
Type	TP	FN	FP	TP	TN	FP	TP	TP	TP	FN

TABLE 2.10 Confusion matrix for sentiment classification of positive (1) and negative (−1) sentiments for ten sentences. We constructed this from expected labels $y = [1, 1, -1, 1, -1, -1, 1, 1, 1, 1]$ and predicted labels $\hat{y} = [1, -1, 1, 1, -1, 1, 1, 1, 1, -1]$. The tabulations follow from mapping in Table 2.9.

	Predicted	
Actual	**Positive**	**Negative**
Positive	5 (TP)	2 (FN)
Negative	2 (FP)	1 (TN)

Word Embedding

Word embedding is a technique in Natural Language Processing (NLP) that represents words as numerical vectors (lists of numbers) in a continuous vector space, where similar words have similar vector representations.

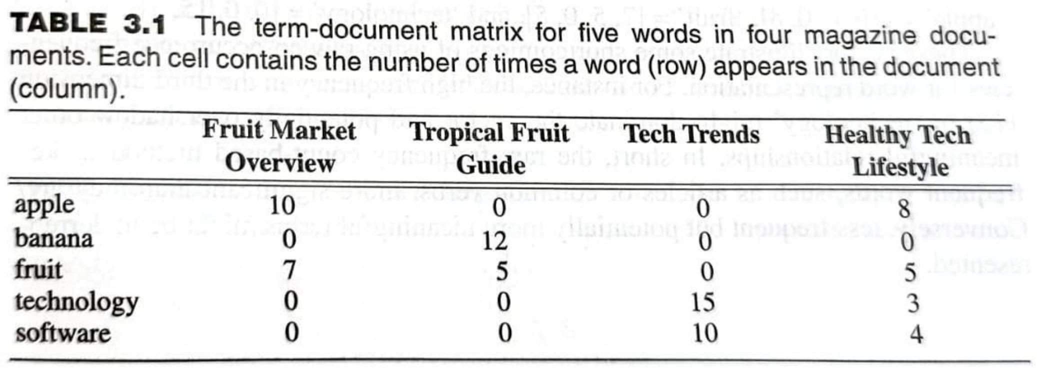


TABLE 3.1 The term-document matrix for five words in four magazine documents. Each cell contains the number of times a word (row) appears in the document (column).

	Fruit Market Overview	Tropical Fruit Guide	Tech Trends	Healthy Tech Lifestyle
apple	10	0	0	8
banana	0	12	0	0
fruit	7	5	0	5
technology	0	0	15	3
software	0	0	10	4

Example 3.1 Let us determine whether 'fruit' is more similar to 'apple' or 'technology' using the example from Table 3.1.

The vector representations for the words are: 'apple' = [10, 0, 0, 8], 'fruit' = [7, 5, 0, 5], and 'technology' = [0, 0, 15, 3]. Using Equation (3.4), we calculate the cosine similarity by:

$$\begin{aligned}\cos(apple, fruit) &= \frac{apple \cdot fruit}{\| apple \| \, \| fruit \|} \\ &= \frac{(10\times 7)+(0\times 5)+(0\times 0)+(8\times 5)}{\sqrt{10^2+0^2+0^2+8^2}\,\sqrt{7^2+5^2+0^2+5^2}} \\ &= \frac{100}{12.81\times 9.95} \\ &= 0.862\end{aligned}$$

$$\begin{aligned}\cos(technology, fruit) &= \frac{technology \cdot fruit}{\| technology \| \, \| fruit \|} \\ &= \frac{(0\times 7)+(0\times 5)+(15\times 0)+(3\times 5)}{\sqrt{0^2+0^2+15^2+3^2}\,\sqrt{7^2+5^2+0^2+5^2}} \\ &= \frac{15}{15.30\times 9.95} \\ &= 0.098\end{aligned}$$

These results indicate that in this vector space representation, 'fruit' shows a significantly higher semantic similarity to 'apple' than to 'technology'. This outcome aligns with the intuitive conceptual relationships among these terms.

Term Frequency-Inverse Document Frequency

The co-occurrence matrices discussed above use raw frequency counts to represent relationships between words and documents or between words and other words. However, this approach has significant limitations. For example, consider the vectors: 'apple' = [10, 0, 0, 8], 'fruit' = [7, 5, 0, 5], and 'technology' = [0, 0, 15, 3].

These vectors illustrate some shortcomings of using raw co-occurrence frequencies for word representation. For instance, the high frequency in the third dimension (15) of 'technology' might dominate the vector and potentially overshadow other meaningful relationships. In short, the raw frequency count-based method makes frequent words, such as articles or common verbs, more significant than they are. Conversely, less frequent but potentially more meaningful terms might be underrepresented.

To address these limitations, we introduce an approach called *Term Frequency-Inverse Document Frequency* (TF-IDF). TF-IDF assigns weight to a word in a document based on its statistical significance within the document and across a collection of documents or corpus. It consists of *Term Frequency* (TF) and *Inverse Document Frequency* (IDF).

1. **Term Frequency (TF):** TF measures how frequently a term occurs in a document. It is typically calculated as:

$$TF_{t,d} = f_{t,d} \tag{3.5}$$

where $f_{t,d}$ is the raw frequency of term t in document d. Since the number of documents can be very large, a logarithmic scaling can be used to dampen the effect of large frequency differences:

$$TF_{t,d} = \begin{cases} 1 + \log(f_{t,d}), & \text{if } f_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

2. **Inverse Document Frequency (IDF):** IDF measures how important a term is across the entire corpus. It is calculated as:

$$\mathrm{IDF}_t = \log\left(\frac{N}{n_t}\right)$$

where N is the total number of documents in the corpus, and n_t is the number of documents containing the term t.

TF-IDF Calculation. TF-IDF is a weighted score that combines the local importance of a term within a document (TF) with its global significance across the corpus (IDF). This statistical measure aims to reflect how important a word is to a document in a collection or corpus. TF-IDF is calculated by multiplying the TF and the IDF of a term:

$$\text{TF-IDF}_{t,d,D} = \mathrm{TF}_{t,d} \times \mathrm{IDF}_{t,D}$$

Here, $\text{TF-IDF}_{t,d,D}$ is the TF-IDF of term t in document d over the corpus D. A higher TF-IDF score for a term signifies that it is both frequent within the document and rare enough in the corpus, thus making it more relevant to the document's content. TF-IDF balances the frequency of a term in a specific document with its rarity across all documents, providing a more nuanced measure of term importance than raw frequency alone.

Vector Representation Using TF-IDF. The TF-IDF score of each term or vocabulary word in the term-document matrix can also be used to represent a document or the word. Consider the following example:

Example 3.2 Consider a small corpus of 4 documents:

1. Doc 1: '*The quick brown fox jumps over the lazy dog*'.
2. Doc 2: '*The lazy dog sleeps all day*'.
3. Doc 3: '*The quick brown fox hunts in the forest*'.
4. Doc 4: '*A lazy afternoon in the forest*'.

Let us consider these four documents and the terms[1] 'the', 'lazy', 'dog', 'quick', and 'fox'. In Table 3.2, each value is the TF score. For example, the TF for '*lazy*' with respect to Doc 2 in Table 3.2 is computed by: *TF*('*lazy*', Doc 2) = *count*('*lazy*', Doc 2) = 1. Next, we compute the IDF score for each term *q*:

- IDF('the') = $\log_2(4/3) \approx 0.415$
- IDF('lazy') = $\log_2(4/3) \approx 0.415$
- IDF('dog') = $\log_2(4/2) = 1$
- IDF('quick') = $\log_2(4/2) = 1$
- IDF('fox') = $\log_2(4/2) = 1$

We then calculate the TF-IDF score for each term. We do this by multiplying each TF value by its corresponding IDF, as presented in Table 3.3. For instance, in Table 3.3, the TF-IDF value for the term 'lazy' in Doc 1 is calculated as:

$$\begin{aligned}\text{TF-IDF('lazy', Doc 1)} &= \text{TF('lazy', Doc 1)} \times \text{IDF('lazy')} \\ &= 1 \times 0.415 = 0.415\end{aligned}$$

TABLE 3.2 The TF matrix for terms 'the', 'lazy', 'dog', 'quick', and 'fox' with respect to four documents: Doc 1, Doc 2, Doc 3, and Doc 4.

	the	lazy	dog	quick	fox
Doc 1	2	1	1	1	1
Doc 2	1	1	1	0	0
Doc 3	2	0	0	1	1
Doc 4	1	1	0	0	0

TABLE 3.3 The TF-IDF matrix for the terms, 'the', 'lazy', 'dog', 'quick', and 'fox' with respect to the four documents: Doc 1, Doc 2, Doc 3, and Doc 4.

	the	lazy	dog	quick	fox
Doc 1	0.830	0.415	1	1	1
Doc 2	0.415	0.415	1	0	0
Doc 3	0.830	0	0	1	1
Doc 4	0.415	0.415	0	0	0

Each row vector in Table 3.3 represents the corresponding document. For instance, Doc 2 is represented as [0.415, 0.415, 1, 0, 0]. Each column vector can be used to represent the word vector. For example, the word 'quick' is represented as [1, 0, 1, 0].