Natural Language Processing with Pytorch

Build Intelligent Language Applications Using Deep Learning

Stephen CUI

April 20, 2023

Chapter 1

Introduction

1.1 The Supervised Learning Paradigm

We can break down the supervised learning paradigm, as illustrated in Figure 1.1, to six main concepts:

Observations Observations are items about which we want to predict something. We denote observations using x. We sometimes refer to the observations as inputs.

Targets Targets are labels corresponding to an observation. These are usually the things being predicted. Following standard notations in machine learning/deep learning, we use y to refer to these. Sometimes, these labels known as the *ground truth*.

Model A model is a mathematical expression or a function that takes an observation, x, and predicts the value of its target label.

Parameters Sometimes also called weights, these parameterize the model. It is standard to use the notation w (for weights) or $\hat{w}$.

Predictions Predictions, also called estimates, are the values of the targets guessed by the model, given the observations. We denote these using a "hat" notation. So, the prediction of a target y is denoted as $\hat{y}$.

Loss function A loss function is a function that compares how far off a prediction is from its target for observations in the training data. Given a target and its prediction, the loss function assigns a scalar real value called the loss. The lower the value of the loss, the better the model is at predicting the target. We use L to denote the loss function.

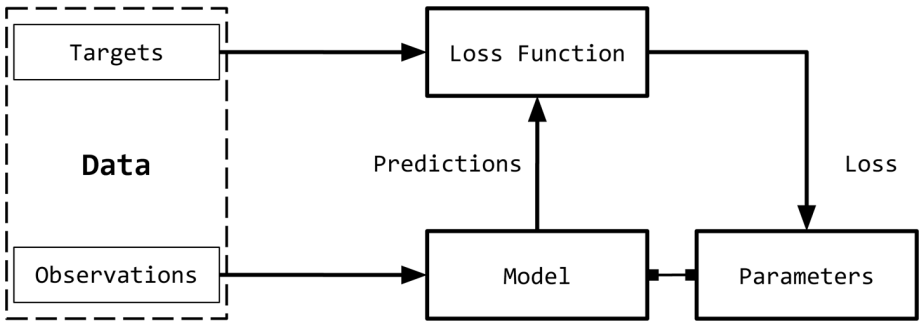

Figure 1.1: The supervised learning paradigm, a conceptual framework for learning from labeled input data.

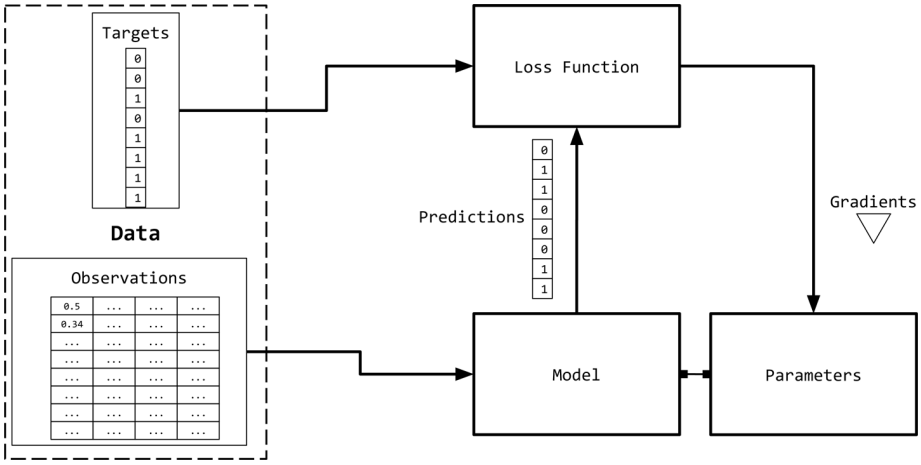

Figure 1.2: Observation and target encoding: The targets and observations from Figure 1.1 are represented numerically as vectors, or tensors. This is collectively known as input "encoding."

Consider a dataset $D = \{X_i, y_i\}_{i=1}^{n}$ with n examples. Given this dataset, we want to learn a function (a model) f parameterized by weights w. That is, we make an assumption about the structure of f, and given that structure, the learned values of the weights w will fully characterize the model. For a given input X, the model predicts $\hat{y}$ as the target:

$$\hat{y} = f(X, w)$$

In supervised learning, for training examples, we know the true target y for an observation. The loss for this instance will then be $L(y, \hat{y})$. Supervised learning then becomes a process of finding the optimal parameters/weights w that will minimize the cumulative loss for all the n examples.

1.2 Observation and Target Encoding

We will need to represent the observations (text) numerically to use them in conjunction with machine learning algorithms. Figure 1.2 presents a visual depiction.

1.2.1 One-Hot Representation

The one-hot representation, as the name suggests, starts with a zero vector, and sets as 1 the corresponding entry in the vector if the word is present in the sentence or document. we use 1_w to mean one-hot representation for a token(词元)/word w.

Using the encoding shown in Figure 1-3, the one-hot representation for the phrase "like a banana" will be a 3×8 matrix, where the columns are the eight-dimensional one-hot vectors. It is also common to see a "collapsed" or a binary encoding where the text/phrase is represented by a vector the length of the vocabulary, with 0s and 1s to indicate absence or presence of a word. The binary encoding for "like a banana" would then be: [0, 0, 0, 1, 1, 0, 0, 1].

1.2.2 Term-Frequency(TF) Representation

The TF representation of a phrase, sentence, or document is simply the sum of the one-hot representations of its constituent words. We denote the TF of a word w by $TF(w)$.

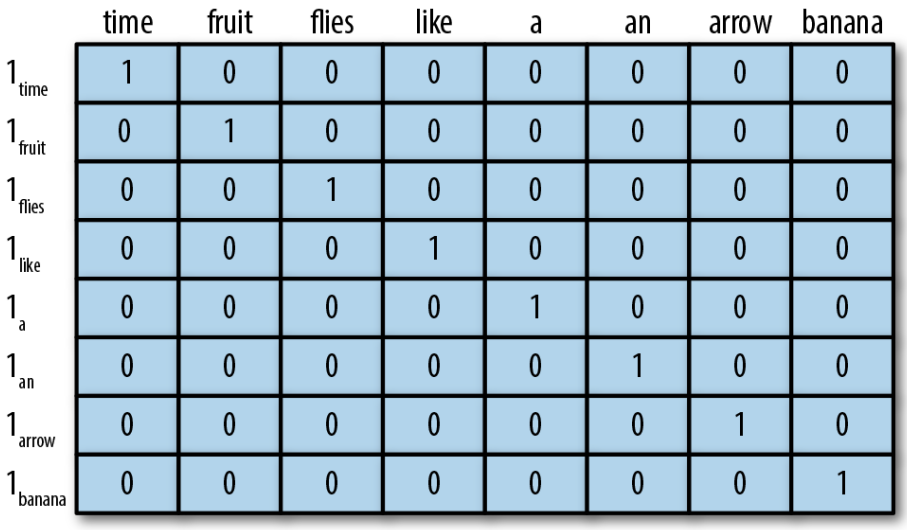

Figure 1.3: One-hot representation for encoding the sentences "Time flies like an arrow" and "Fruit flies like a banana."

1.2.3 TF-IDF Representation

The IDF representation penalizes common tokens and rewards rare tokens in the vector representation. The $IDF(w)$ of a token w is defined with respect to a corpus as:

$$IDF(w) = \log \frac{N}{n_w} \tag{1.1}$$

where n_w is the number of documents containing the word w and N is the total number of documents. The TF-IDF score is simply the product $TF(w) * IDF(w)$.

1.2.4 Target Encoding

Many NLP tasks actually use categorical labels, wherein the model must predict one of a fixed set of labels.

Some NLP problems involve predicting a numerical value from a given text. For example, given an English essay, we might need to assign a numeric grade or a readability score.

1.3 Computational Graphs

Technically, a computational graph is an abstraction that models mathematical expressions.

Inference (or prediction) is simply expression evaluation (a forward flow on a computational graph).

1.4 PyTorch Basics

Dynamic Versus Static Computational Graphs

Static frameworks like Theano, Caffe, and TensorFlow require the computational graph to be first declared, compiled, and then executed.6 Although this leads to extremely efficient implementations (useful in production and mobile settings), it can become quite cumbersome during research and development. Modern frameworks like Chainer, DyNet, and PyTorch implement dynamic computational graphs to allow for a more flexible, imperative style of development, without needing to compile the models before every execution. Dynamic computational graphs are especially useful in modeling NLP tasks for which each input could potentially result in a different graph structure.

At the core of the library is the tensor, which is a mathematical object holding some multidimensional data.

1.4.1 Creating Tensors

Any PyTorch method with an underscore (_) refers to an in-place operation; that is, it modifies the content in place without creating a new object.

1.4.2 Tensor Types and Size

Each tensor has an associated type and size. The default tensor type when you use the torch.Tensor constructor is torch.FloatTensor. There are two ways to specify the initialization type: either by directly calling the constructor of a specific tensor type, such as `FloatTensor` or `LongTensor`, or using a special method, `torch.tensor()`, and providing the dtype.

1.4.3 Tensors and Computational Graphs

“Bookkeeping operations”是深度学习中的一个术语，它通常用于指代在神经网络训练过程中的一系列轻量级操作，用于跟踪和记录训练期间的各种统计数据。

这些操作包括记录训练损失、计算梯度、更新模型参数等。这些操作通常被认为是“轻量级”的，因为它们不涉及任何繁重的计算，而是在训练过程中进行简单的统计计算和数据更新。

在深度学习中，由于神经网络的复杂性，训练过程通常需要进行大量的迭代和调整，而这些“轻量级”的操作可以帮助我们跟踪和调整训练过程中的各种参数和指标，从而更好地优化模型的性能。

When you create a tensor with `requires_grad=True`, you are requiring PyTorch to manage bookkeeping information that computes gradients. First, PyTorch will keep track of the values of the forward pass. Then, at the end of the computations, a single scalar is used to compute a backward pass. The backward pass is initiated by using the `backward()` method on a tensor resulting from the evaluation of a loss function. The backward pass computes a gradient value for a tensor object that participated in the forward pass.

1.4.4 CUDA Tensors

Keep in mind that it is expensive to move data back and forth from the GPU. Therefore, the typical procedure involves doing many of the parallelizable computations on the GPU and then transferring just the final result back to the CPU.

Chapter 2

A Quick Tour of Traditional NLP

Natural language processing (NLP) and *computational linguistics* (CL) are two areas of computational study of human language.

2.1 Corpora, Tokens, and Types

All NLP methods, be they classic or modern, begin with a text dataset, also called a corpus (plural: corpora). A corpus usually contains raw text (in ASCII or UTF-8) and any metadata associated with the text. The raw text is a sequence of characters(bytes), but most times it is useful to group those characters into contiguous units called tokens. In English, tokens correspond to words and numeric sequences separated by white-space characters or punctuation.

The metadata could be any auxiliary piece of information associated with the text, like identifiers, labels, and timestamps. In machine learning parlance, the text along with its metadata is called an instance or data point. The corpus (Figure 2.1), a collection of instances, is also known as a dataset.

The process of breaking a text down into tokens is called tokenization.

Types are unique tokens present in a corpus. The set of all types in a corpus is its vocabulary or lexicon. Words can be distinguished as content words and stopwords.

2.2 Unigrams, Bigrams, Trigrams,. . ., N-grams

N-grams are fixed-length (n) consecutive token sequences occurring in the text.

2.3 Lemmas and Stems

Lemmas are root forms of words. Sometimes, it might be useful to reduce the tokens to their lemmas to keep the dimensionality of the vector representation low. This reduction is called lemmatization. Stemming is the poor-man's lemmatization.3 It involves the use of handcrafted rules to strip endings of words to reduce them to a common form called stems.

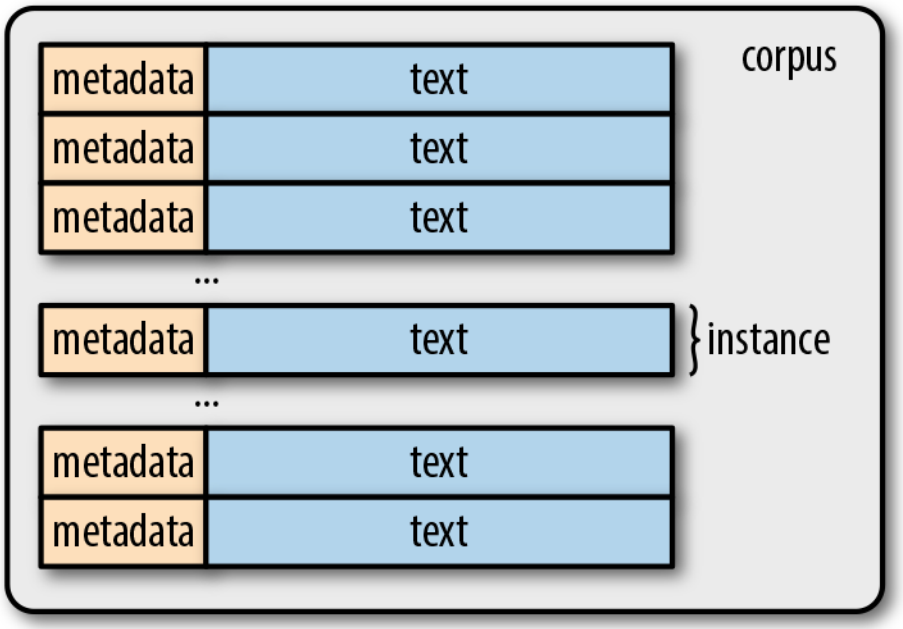

Figure 2.1: The corpus: the starting point of NLP tasks.

2.4 Categorizing Sentences and Documents

2.4.1 Categorizing Words: POS Tagging

We can extend the concept of labeling from documents to individual words or tokens. A common example of categorizing words is part-of-speech (POS) tagging.

2.4.2 Categorizing Spans: Chunking and Named Entity Recognition

Often, we need to label a span of text; that is, a contiguous multitoken boundary. For example, consider the sentence, "Mary slapped the green witch." We might want to identify the noun phrases (NP) and verb phrases (VP) in it, as shown here:

```
[NP Mary] [VP slapped] [the green witch].
```

This is called chunking or shallow parsing. Shallow parsing aims to derive higherorder units composed of the grammatical atoms, like nouns, verbs, adjectives, and so on.

Another type of span that's useful is the **named entity**. A named entity is a string mention of a real-world concept like a person, location, organization, drug name, and so on.

Structure of Sentences

Whereas shallow parsing identifies phrasal units, the task of identifying the relationship between them is called **parsing**.

Parse trees indicate how different grammatical units in a sentence are related hierarchically. The parse tree in Figure 2.2 shows what's called a constituent parse. Another, possibly more useful, way to show relationships is using dependency parsing, depicted in Figure 2.3.

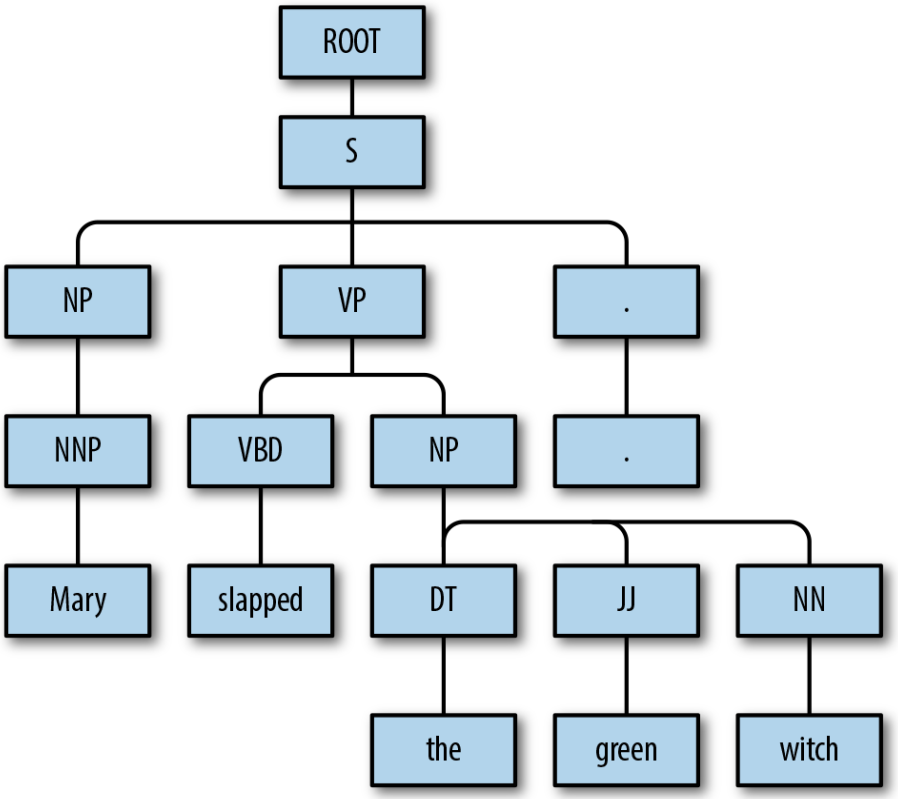

Figure 2.2: A constituent parse of the sentence “Mary slapped the green witch.”

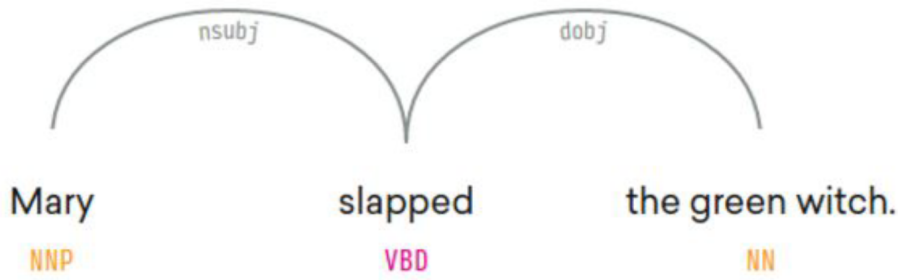

Figure 2.3: A dependency parse of the sentence “Mary slapped the green witch.”

2.5 Word Senses and Semantics

Words have meanings, and often more than one. The different meanings of a word are called its senses.

Chapter 3

Foundational Components of Neural Networks

3.1 The Perceptron: The Simplest Neural Network

Each perceptron unit has an input (x), an output (y), and three "knobs": a set of weights (w), a bias (b), and an activation function (f). The weights and the bias are learned from the data, and the activation function is handpicked depending on the network designer's intuition of the network and its target outputs. Mathematically, we can express this as follows:

$$y = f(\mathbf{w}x + \mathbf{b})$$

t is usually the case that there is more than one input to the perceptron. We can represent this general case using vectors. That is, $\mathbf{x}$, and $\mathbf{w}$ are vectors, and the product of $\mathbf{w}$ and $\mathbf{x}$ is replaced with a dot product:

$$y = f(\mathbf{w}\mathbf{x} + \mathbf{b})$$

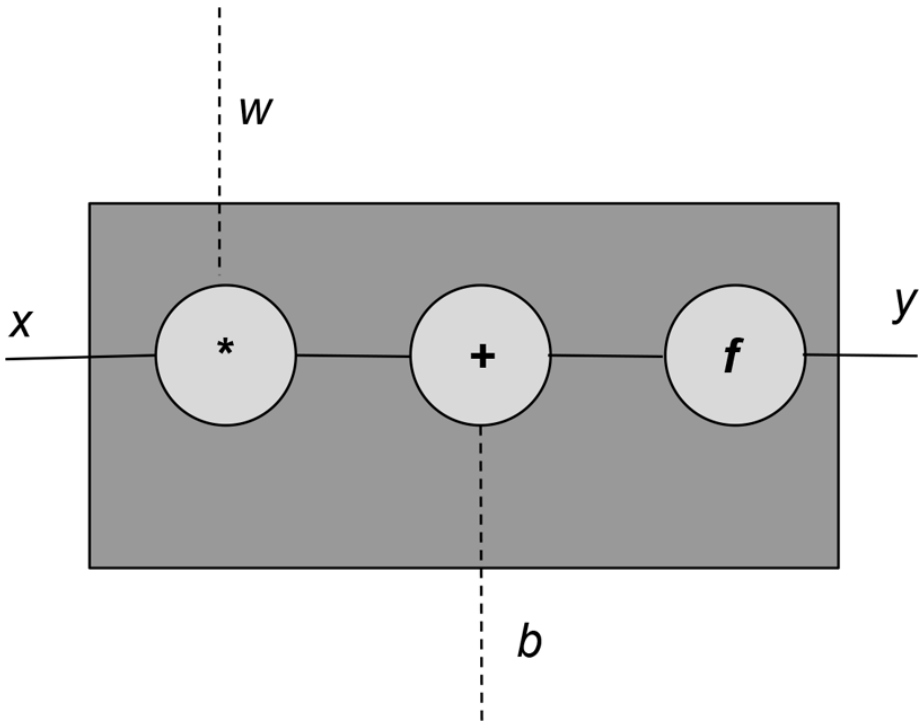

Figure 3.1: The computational graph for a perceptron with an input (x) and an output(y). The weights (w) and bias (b) constitute the parameters of the model.