目录

Chapter 1

1.1

1.1.1 Estimating class probabilities in multiclass classification via the softmax function

The softmax function is a soft form of the argmax function; instead of giving a single class index, it provides the probability of each class. Therefore, it allows us to compute meaningful class probabilities in multiclass settings (multinomial logistic regression).

In softmax, the probability of a particular sample with net input z belonging to the ith class can be computed with a normalization term in the denominator, that is, the sum of the exponentially weighted linear functions:

$$p(z) = \sigma(z) = \frac{e^{z_i}}{\sum_{j=1}^{M} e^{z_j}} \tag{1.1}$$

1.1.2 Broadening the output spectrum using a hyperbolic tangent

Another sigmoidal function that is often used in the hidden layers of artificial NNs is the hyperbolic tangent (commonly known as tanh), which can be interpreted as a rescaled version of the logistic function:

$$\begin{aligned} \sigma_{logistic}(z) &= \frac{1}{1+e^{-z}} \\ \sigma_{tanh}(z) &= 2 \times \sigma_{logistic}(2z) - 1 = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}} \end{aligned} \tag{1.2}$$

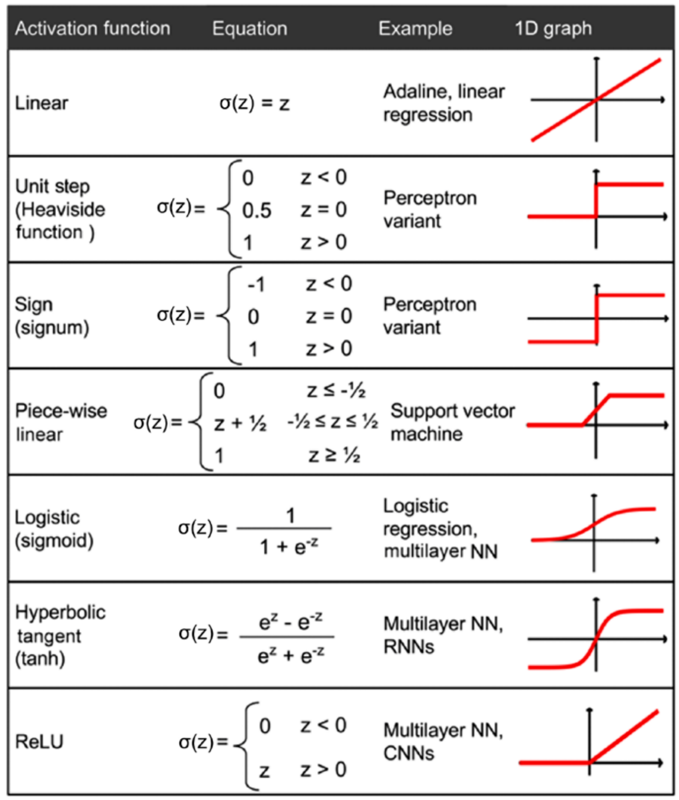

图 1.1: The activation functions covered

The advantage of the hyperbolic tangent over the logistic function is that it has a broader output spectrum ranging in the open interval $(-1, 1)$, which can improve the convergence of the backpropagation algorithm.

Note that using torch.sigmoid(x) produces results that are equivalent to torch.nn.Sigmoid()(x). torch.nn.Sigmoid is a class to which you can pass in parameters to construct an object in order to control the behavior. In contrast, torch.sigmoid is a function.

Chapter 2

Classifying Images with Deep Convolutional Neural Networks

2.1 The building blocks of CNNs

2.1.1 Discrete convolutions in one dimension

Determining the size of the convolution output

The output size of a convolution is determined by the total number of times that we shift the filter, $\boldsymbol{w}$, along the input vector. Let's assume that the input vector is of size n and the filter is of size m. Then, the size of the output resulting from $\boldsymbol{y} = \boldsymbol{x} * \boldsymbol{w}$, with padding p and stride s, would be determined as follows:

$$o = \left\lfloor \frac{n + 2p - m}{s} \right\rfloor + 1 \tag{2.1}$$

2.1.2 Subsampling layers

Subsampling is typically applied in two forms of pooling operations in CNNs: max-pooling and mean-pooling (also known as average-pooling). The pooling layer is usually denoted by $P_{n_1 \times n_2}$.

The advantage of pooling is twofold:

- Pooling (max-pooling) introduces a local invariance. This means that small changes in a local neighborhood do not change the result of max-pooling. Therefore, it helps with generating features that are more robust to noise in the input data.

- Pooling decreases the size of features, which results in higher computational efficiency. Furthermore, reducing the number of features may reduce the degree of overfitting as well.

Overlapping versus non-overlapping pooling

Traditionally, pooling is assumed to be non-overlapping. Pooling is typically performed on non-overlapping neighborhoods, which can be done by setting the stride parameter equal to the pooling size. For example, a non-overlapping pooling layer, $P_{n_1 \times n_2}$, requires a stride parameter $s = (n_1, n_2)$. On the other hand, overlapping pooling occurs if the stride is smaller than the pooling size.