CS 189 Introduction to Machine Learning
Fall 2017 DIS13

1 Convolution and Backprop Revisited

In this problem, we will explore how image filtering can help us create useful high-level features that we can use instead of raw pixel values. We will walk through how discrete 2D convolution works and how we can use the backprop algorithm to compute derivatives through this operation.

(a) To start, let's consider convolution in one dimension. Convolution can be viewed as a function that takes a signal $f[t]$ and a filter $g[t]$, and the discrete convolution at point t of the signal with the filter is given as

$$(f * g)[t] = \sum_{k=-\infty}^{\infty} f[k]g[t-k]$$

If the filter $g[t]$ is nonzero in a finite range, then the summation can be reduce to just the range in which the filter is nonzero, which makes computing a convolution on a computer possible.

As an example, we can use convolution to compute a derivative approximation with finite differences. The derivative approximation of the signal is given by $f'[t] \approx (f[t+1] - f[t-1])/2$. Design a filter $g[t]$ such that $(f * g)[t] = f'[t]$.

(b) Convolution in two dimensions is similar to the one dimensional case except that we have an additional dimension to sum over. If we have some signal $I[x,y]$ and some filter $G[x,y]$, then the convolution at the point (x,y) is given by

$$(I * G)[x,y] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} I[m,n]G[x-m,y-n]$$

or equivalently,

$$(I * G)[x,y] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} G[m,n]I[x-m,y-n]$$

because convolution is commutative.

For implementation, we'll have an image I, which has three color channels I_r, I_g, I_b each of size $W \times H$, where W is the image width, H is the height. Each color channel represents the intensity of red, green, and blue for each pixel in the image. We also have the filter G with finite support. The filter also has three color channels, G_r, G_g, G_b and we represent these as

a $w \times h$ matrix where w and h are the width and height of the filter. The output $(I * G)[x, y]$ at point (x, y) is given by

$$(I * G)[x, y] = \sum_{a=1}^{w} \sum_{b=1}^{h} \sum_{c \in \{r,g,b\}} I_c[x + a, y + b] \cdot G_c[a, b]$$

In this case, the size of the output will be $(1 + W - w) \times (1 + H - h)$, and we only evaluate the convolution within the image I. We will not consider how to compute the convolution along the edge of the image. To reduce the dimension of the output, we can do a strided convolution in which we sample the convolution at every s point instead of every point. The resulting output will be $(1 + (W - w)/s) \times (1 + (W - w)/s)$.

Write pseudocode to compute the convolution of an image I with a set of filters G and a stride of s.

(c) Filters can be used to identify different types of features in an image such as edges or corners. Design a filter G that outputs a large value for vertically oriented edges in image I.

(d) Although hand-crafted filters can produce good results, we can learn filters specific to the problem that we are solving. To learn filters, we need to take derivatives of the filter parameters with respect to the error function. These derivatives can be computed efficiently using the backprop algorithm similar to how we computed derivates for model parameters in dense layers.

Given the output of a convolution L, the derivative of the error function with respect to the output ∂L, the convolution filters G, and the input image I, write an expression for the derivative of the filter parameter $G_c[i, j]$ for $c \in \{r, g, b\}$ and the derivative of $I_c[x, y]$ from the input image.

(e) Sometimes, the output of a convolution can be large, and we might want to reduce the dimension of this. A common method to reduce the dimension of an image is called max pooling. This method works similar to convolution in that we have a filter that moves around the image, but instead of multiplying the filter with a subsection of the image, we take the maximum value in the subimage. Max pooling can also be thought of as downsampling the image but keeping the largest activations for each channel from the original input. Given a filter size of $w \times h$, and a stride s, the output will be $(1 + (W - w)/s) \times (1 + (H - h)/s)$ for an input image of size $W \times H$.

Let the output of a max pooling operation be L, write an expression for element $L[i, j]$ of the output.

(f) Explain how we can use the backprop algorithm to compute derivates through the max pooling operation.