

1. Maximal Update Learning Rates During Training

Assume we are using a minibatch size of 1. For simplicity, consider a neural network layer with input $\mathbf{x}_k \in \mathbb{R}^{d_{\text{in}}}$ that is sampled from an i.i.d. unit Gaussian, and weights $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$.

- (a) First, compute the (stochastic) gradient $\nabla_W \mathcal{L}$ for $\mathbf{y} = W\mathbf{x} + \mathbf{b}$ and downstream loss function $\mathcal{L}$ where the loss depends on $\mathbf{y}$. Your answer should be in terms of $\mathbf{x}_k$ and $\mathbf{g}_k = \nabla_{\mathbf{y}} \mathcal{L}$, where the subscript k is the index of the example in the batch.

- (b) In SignGD, we know that W is updated as below:

$$W_{t+1} \leftarrow W_t + \eta \text{sign}(\nabla_W \mathcal{L}(\mathbf{y})) .$$

What is the expected RMS norm squared of the change in features $\Delta \mathbf{y} = \eta \text{sign}(\nabla_W \mathcal{L}(\mathbf{y}))\mathbf{x}_k$? How does this scale with d_{out} or d_{in} ? What constant should we multiply the update by to ensure that the expected RMS norm squared of $\Delta \mathbf{y}$ does not depend on either d_{out} or d_{in} ?

2. Weight Sharing in CNN

In this question we will look at the mechanism of weight sharing in convolutions. Let's start with a 1-dimension example. Suppose that we have a 9 dimensional input vector and compute a 1D convolution with the kernel filter that has 3 weights (parameters).

$$\mathbf{k} = \begin{bmatrix} k_1 & k_2 & k_3 \end{bmatrix}^T, \mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & \dots & x_9 \end{bmatrix}^T$$

- (a) What's the **output dimension** if we apply filter $\mathbf{k}$ with no padding and stride of 1? What's the **first element** of the output? What's the **last element**?
- (b) What's the **output dimension** if we apply $\mathbf{k}$ with "same padding" (padding $\mathbf{x}$ with zeros) and stride of 1? What's the **first element** of the output? What's the **last element**?
- (c) Convolution is a linear operation and we can express it in the form of linear layers, ie. $\mathbf{h} = \mathbf{K}\mathbf{x}$ (assume the bias term is zero). Recall that in lecture, we discussed that CNN filters have the property of **weight sharing**, meaning that different portions of an image can share the same weight to extract the same set of features.
Find $\mathbf{K}$, the linear transformation matrix corresponding to the convolution applied in part (a).
(Hint: What is the dimension of $\mathbf{K}$?)

- (d) **Find K for part (b), where we apply “same padding”.** (*Hint: You should only be adding rows to your answer in part c.*)
- (e) Suppose that we no longer used weight sharing: for each location within the input, we apply a different filter. **How does this change our K matrix? How many parameters do we have now?**
- (f) We want to know the general formula for computing the output dimension of a convolution operation. Suppose we have an input vector of dimension W_{in} and a filter of size K . If we assume stride of 1 and no padding, **what’s the output dimension W_{out} ? If we applied stride of s and padding of size p , how would the dimension change?**

- (g) Now, consider a convolution over a 2-dimensional image using a 2-dimensional filter. If our input image is square with dimension $W_{in} \times W_{in}$ and we apply square filter with dimension $K \times K$, stride 1, and no padding, **what are the dimensions of the resulting output? If we applied stride s and padding p , what would the output dimension be?**

Let's take what we've learned into actual applications on image tasks. Suppose our input is a 256×256 RGB image. You are tasked to design a convolution layer that takes in 256×256 images as input and outputs a 32-channel feature map using convolution kernels each with a kernel size of 5.

- (h) Conventionally in frameworks such as PyTorch, images are arranged in 3-D tensors of shape `[channels, height, width]`. Our convolutional layer takes in the image as input and outputs a feature map arranged in 3-D tensors of shape `[32, height, width]`. **What is the shape of our input tensor (fill in the numerical value)? What is the shape of the tensor that stores all the convolution kernels in the convolution layer?**

(Hint: the convolution kernel tensor contains a convolution kernel for each pair of input channel and output channel.)

- (i) Now apply convolution on our image tensor with no padding and stride of 2. **What is the output tensor's dimension? Considering all kernel filters, how many parameters do we have?**

If instead of using a convolution layer, we use a fully connected layer (flattening the image to treat it as a single vector and sized so that the output vector has the same number of activations as the output of the convolutional layer), **how many parameters does the fully connected layer contain?** Feel free to use a calculator for this question.

Contributors: Kevin Frans, Anant Sahai, Gireeja Ranade, Matteo Guarrera, Mert Cemri, Sizhe Chen, Jake Austin, Suhong Moon, Kevin Li.