Due Date : March 9th, 2018

Instructions

- *For all questions, show your work!*
- *This part (theory) is to be done individually.*
- *Use a document preparation system such as LaTeX. We will also accept scans of exceptionally clean hand-written answers – anything illegible will be considered an error.*
- *Submit your answers electronically via the course studium page.*

1. **(5 points) Convolutions**
   Compute the *full*, *valid*, and *same* convolution (with kernel flipping) for the following 1D matrices : $\begin{bmatrix}1, & 2, & 3, & 4\end{bmatrix} * \begin{bmatrix}1, & 0, & 2\end{bmatrix}$

2. **(5 point) Convolutional Neural Networks**
   Consider a 3-layer CNN. Assume the input is a colorful image of size 256×256 in the RGB representation. The first layer convolves 64 8×8 kernels with the input, using a stride of 2 and no padding. The second layer subsamples the output of the first layer with a 5×5 non-overlapping max pooling. The third layer convolves 128 4×4 kernels with a stride of 1 and a zero-padding of size 1 on each border.
   (a) What will be the size of the output of the last layer ? Answer with a simple scalar.
   (b) Without considering the biases, how many parameters are needed for the last convolution ?

3. **(10 points) Kernel configuration for CNNs**
   In this question, we will focus on the ability to understand how kernel sizes vary based on the different parameters. Let's assume squared inputs for simplicity. Furthermore, assume we are working on images of color (RGB) of size $3 \times 64 \times 64$ pixels. The notation used for inputs/outputs is ($channels \times x \times y$)
   (a) For the first convolutional layer, the input shape is $3 \times 64 \times 64$ and the output shape is $64 \times 32 \times 32$. Provide a configuration of the convolution operation that would match the input/output shapes (i.e. provide the kernel, stride, padding and dilatation between kernel elements).
      i. Assuming no dilatation and kernel size of 8×8.
      ii. Assuming dilatation of 6 between kernel elements and stride of 2.
   (b) For the second pooling layer, the input shape is $64 \times 32 \times 32$ and the output shape is $64 \times 8 \times 8$. Provide a configuration of the pooling operation that would match the input/output shapes, assuming no overlapping of pooling windows and no padding (i.e. what kernel size and stride to use).
   (c) Without any padding and using the input from question 2 ($64 \times 32 \times 32$), what would have been the output size should the kernel be of size 8×8, but with the stride 4×4 ?

(d) In the last convolutional layer, the input shape is $64 \times 8 \times 8$ and the output shape is $128 \times 4 \times 4$. Provide a configuration of the convolution operation that would match the input/output shapes (i.e. provide the kernel, stride, padding and dilatation between kernel elements).

i. Assuming no padding and no dilatation.

ii. Assuming dilatation of 1 and padding of 2.

iii. Assuming padding of 1 and no dilatation.

4. **(10 point) Dropout as weight decay**
Consider a linear regression problem with input data $X \in \mathbb{R}^{n \times d}$. weights $w \in \mathbb{R}^{d \times 1}$ and and targets $y \in \mathbb{R}^{n \times 1}$. Now, suppose that dropout is being applied to the input units with probability p.

(a) Rewrite the input data matrix taking into account the probability of each unit to be dropped out (*Hint : the probability of each unit to be dropped out is a Bernoulli random variable with probability* p).

(b) What is the cost function of the linear regression with dropout ?

(c) Show that applying dropout to the linear regression problem aforementioned can be seen as using L2 regularization in the loss function. Specifically, let the objective function be reparameterized by the rescaled parameter $\bar{w} = pw$, show the the solution $\bar{w}^*$ to the objective is given by

$$\bar{w}^* = (X^T X + \lambda \Gamma^2)^{-1} X^T y$$

where λ is a function of p and $\Gamma = \text{diag}(X^T X)^{\frac{1}{2}}$. How does this relate to the solution to the L2 regularized least square problem ? Explain the difference.

5. **(10 point) Dropout as Geometric Ensemble**
Dropout can be thought of as an efficient bagging method that is composed of an ensemble of neural networks. The *weight scaling rule* of dropout, which indicates that the weights have to be scaled by the inclusion probability prior to inference, allows the dropout as a geometric ensemble to be computationally feasible. Consider the case of a single linear layer model with softmax output

$$P(y = j|v) = \hat{y}_j = \text{softmax}(W^T v + b)_j.$$

where v can be a learned feature or the raw input. Prove that the weight scaling with a factor of 0.5 corresponds exactly to the inference of a conditional probability distribution proportional to the geometric mean over all dropout masks,

$$p_{ens}(y = j|v) \propto \left(\prod_{i=1}^{N} \hat{y}_j^{(i)}, \right)^{\frac{1}{N}}$$

where N is the number of dropout masks, $\hat{y}_j^{(i)} = \text{softmax}(W^T(m_i \odot v) + b)_j$, and m_i is a dropout mask configuration.

6. **(10 point) Normalization**
Weight Normalization (WN) is a reparameterization technique inspired by Batch normalization (BN), aiming at improving the conditioning of the gradient. Instead of having a regular weight parameter (denoted as w) for each neuron, i.e. $y = \sigma(w^T x + b)$, we decouple the weight vector into two terms :

$$w = \frac{g}{\|u\|} u \tag{1}$$

where $g \in \mathbf{R}$ is a scaling factor and u is normalized by the Euclidean $\|u\|$. Doing so has similar effects as implementing BN, but has a lower computational overhead.

(a) Consider the simplest model, where we only have one single output layer conditioned on one input feature x. Assume x is preprocessed to have zero mean and unit variance. A standard BN operation is defined by Equation (8.35) of the deep learning book. Show that in this simple case WN is equivalent to BN (ignoring the learned scale and shift terms) that normalizes the linearly transformed feature $w^T x + b$.

(b) Show that the gradient of a loss function L with respect to the new parameters u can be expressed in the form $sW^* \nabla_w L$, where s is a scalar and W^* is the orthogonal complement projection matrix. As a side note : W^* projects the gradient away from the direction of w, which is usually (empirically) close to a dominant eigenvector of the covariance of the gradient. This helps to condition the landscape over which we want to optimize.

(c) Figure 1 shows the norm of the unnormalized weight parameters (u) used in a MLP with single hidden layer. Data was simulated uniformly at random. Different lines correspond to models trained with different log learning rate, as shown in the legend. Explain the graph. (Hint : Pythagorean theorem and the fact found in the previous question.)

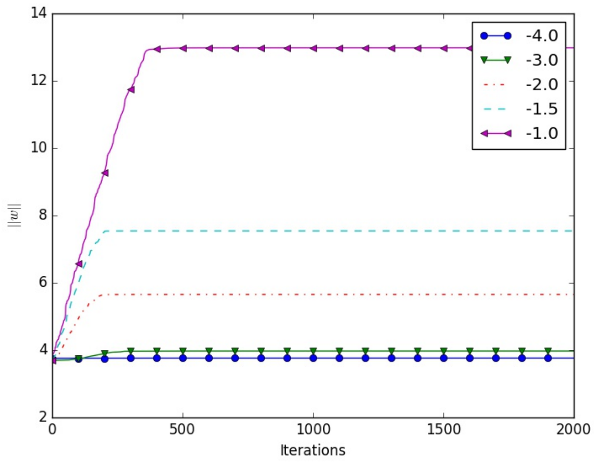

FIGURE 1 – Norm of parameters with different learning rate.