Homework 1

Spring 2019
(Due: Friday September 20, 2019)

Introduction

This assignment is on the foundational aspects of Deep Feedforward Networks, that are covered in Chapter 6 of the recommended text.

Exercises

1. (XOR Function)

   Suppose we have dataset $\mathcal{X} = \{0,1\} \times \{0,1\}$. Recall that the XOR function $f^* : \mathcal{X} \to \{0,1\}$ is defined by

   $$f^*(\boldsymbol{x}) = \begin{cases} 1, & \text{if } x_1 \neq x_2 \\ 0, & \text{otherwise,} \end{cases}$$

   where x_1 and x_2 are the first and second elements of x, respectively.

   a) Is it possible for a linear function in the two-dimensional space $\mathbf{R}^2$ to learn the XOR function? Why?

   b) Is it possible for a feedforward network to correctly learn the XOR function? If no, explain why. If yes, explain in detail a solution using a network with one hidden layer, including the activation function to be used.

2. (RBF Kernel) Recall the radial basis function kernel

   $$K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}, (\boldsymbol{x}, \boldsymbol{x}') \mapsto \exp\{-\gamma \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2\} \tag{1}$$

   Show that the feature space of the RBF kernel corresponds to a dot product in an infinite dimensional vector space, even when $d = 1$ (you can also assume that $\gamma = 1$ for simplicity).

3. (Gradient-Based Learning)

   a) Briefly explain what gradient-based learning is.

   b) What kind of function behavior - resulting from the pairing of output units and cost functions - is suitable for gradient-based learning, and what kinds are not? Please explain with examples.

4. (Cross Entropy)

   Recall that given two probability distributions p and q, the cross-entropy function is defined by

   $$H(p, q) = -\mathbb{E}_{\boldsymbol{x} \sim p}[\log(q(\boldsymbol{x}))] \tag{2}$$

   Moreover, replacing $\boldsymbol{x} \sim p$ by the data distribution $\boldsymbol{x}, \boldsymbol{y} \sim p_{\text{data}}$, and $q(\boldsymbol{x})$ by a model's conditional distribution $p_{\text{model}}(\boldsymbol{y}|\boldsymbol{x})$, we have the usual cross entropy loss function used in machine learning.

a) Derive the cross entropy function from the maximum likelihood principle. More specifically, suppose we are given a set of independent observations $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_n, \boldsymbol{y}_n)\} \subset \mathcal{X} \times \mathcal{Y}$ sampled from p_{data}, where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the target space, and we wish to estimate p_{data} with a parametric model with parameter $\boldsymbol{\theta}$. We carry out this task with maximum likelihood estimation:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmax}}\, p_{\text{model}}(\boldsymbol{y}_1, ..., \boldsymbol{y}_n | \boldsymbol{x}_1, ..., \boldsymbol{x}_n, \boldsymbol{\theta}) \tag{3}$$

Show that, as the number of observations tends to infinity, this optimization problem is equivalent to minimizing the cross entropy loss as described above.

b) What kinds of output units do we expect to work well with the cross entropy loss?

5. (Sigmoid/Softmax)

a) What are the Sigmoid and Softmax functions functions? What is the interpretation of each of them when used as output units to compute an estimate of the conditional probability of the class labels given the input sample?

b) What are the advantages of using the Sigmoid and Softmax output units for gradient-based learning when paired with the cross-entropy loss function?

6. (Linear Activation Function)

Can linear activation functions be useful for the hidden layers of deep feedforward neural networks? Explain why.

7. (ReLU Activation)

a) Explain what the ReLU activation function is.

b) Some immediate generalizations of the ReLU activation function include the absolute value rectification function and the parametric ReLU function. Describe these generalizations and the advantages of using them.

c) What is the maxout activation function? Does it reduce or increase the number of parameters in the network? Why?

8. (Universal Approximation)

a) State the universal approximation theorem.

b) Discuss the generalizations of this theorem that make it apply to the state-of-the-art deep neural networks, and its implications.

c) How does depth of the network affect the number of parameters, generalization, and difficulty of training optimization?

9. (Backpropagation)

a) Suppose we have functions $f : \mathbb{R}^n \to \mathbb{R}^m$, and $g : \mathbb{R}^m \to \mathbb{R}$, and let $h = g \circ f$. What is the gradient $\nabla h(\boldsymbol{x})$ in terms of the Jacobian of f and the gradient ∇g? Illustrate the components of each vector/matrix.

b) Consider a function $J : \mathbb{R}^d \to \mathbb{R}$ explicitly defined by

$$J(u_1, ..., u_d) = f_l \circ f_{l-1} \circ ... \circ f_1(u_1, ..., u_d) \tag{4}$$

where $f_j : \mathbb{R}^{n_j} \to \mathbb{R}^{n_{j+1}}$, and $n_1 = d$ the input dimension, and $n_{l+1} = 1$.

i) Describe, in pseudocode (with a graphical illustration if you wish), how the backpropagation algorithm would compute $\nabla J(u_1, ..., u_d)$, assuming that all intermediate partial derivatives in the computational graph can be stored without regard to memory. Remember to include how the forward passed values are used in the algorithm.

ii) What is the computational complexity of the algorithm in i)? Hint: What computations take place in each step?

c) Now consider a multilayer feedforward network $f : \mathbb{R}^d \to \mathbb{R}^o$ described recursively by

$$\begin{aligned} f(\boldsymbol{x}) =& \boldsymbol{h}^{(l)}, \\ \boldsymbol{h}^{(k)} =& \sigma(\boldsymbol{a}^{(k)}),\ \boldsymbol{a}^{(k)} = \boldsymbol{b}^{(k)} + \boldsymbol{W}^{(k)}\boldsymbol{h}^{(k-1)},\ k \in \{1, ..., l\}, \end{aligned} \tag{5}$$

where σ is the (element-wise) nonlinear activation function, $\boldsymbol{W}^{(k)}$ is the weight matrix of the k-th layer, and $\boldsymbol{b}^{(k)}$ is the bias vector of the k-th layer.

Furthermore, suppose the loss function is described by

$$\mathcal{L} : \mathbb{R}^o \times \mathbb{R}^o \to \mathbb{R}, (\hat{\boldsymbol{y}}, \boldsymbol{y}) \mapsto \mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y}) \tag{6}$$

where $\hat{\boldsymbol{y}} = f(\boldsymbol{x})$ is network output and $\boldsymbol{y}$ is the target output.

i) Describe, in pseudocode, how the backpropagation algorithm would compute the gradient of $\mathcal{L}$ with respect to every trainable parameter in the network. Again, remember to include the use of the forward passed activation values in the algorithm (you can assume ReLU activations for illustration).

ii) Suppose the network f has one hidden layer. Show how the backpropagation algorithm operates by drawing an example computational graph and illustrating the steps.

d) Explain the tradeoff between computation and storage costs in computing the gradient? Where does the basic backpropagation algorithm (For example, Algorithm 6.4 in recommended text) stand with respect to this tradeoff?

10. (Backpropagation Implementation)

a) What are the differences between the Torch and Tensorflow approaches to backpropagation computation? Hint: State the main difference regarding symbolic representations.

b) For the Tensorflow approach, it is important to define a backpropagation method for each operation (op.brop()). Discuss the role of this backpropagation method in computing the gradient, and what it should return. Give an example for either the Matrix Multiplication operation or the ReLU operation.