Machine Learning for Complex Networks
SoSe 2024

Prof. Dr. Ingo Scholtes
Chair of Informatics XV
University of Würzburg

Exercise Sheet 08

Published: June 26, 2024
Due: July 03, 2024
Total points: 10

Please upload your solutions to WueCampus as a scanned document (image format or pdf), a typesetted PDF document, and/or as a `jupyter` notebook.

1. **Computing the Gradients**

   (a) Let $\frac{\partial}{\partial\theta_j}\mathcal{J}(\theta) = \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$ be the gradient for logistic regression. Given a learning rate α, give the gradient descent update formula for θ_j. 1P

   (b) Consider the L2 loss function 2P

   $$L(\beta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

   for a binary perceptron classifier with logistic activation function, parameters $\beta_0, \ldots, \beta_k$ and training data $\vec{x}_i \in \mathbb{R}^k$ for $i = 1, \ldots, n$. Show that the gradient $\nabla L = (\frac{\partial L}{\partial\beta_0}, \ldots, \frac{\partial L}{\partial\beta_k})$ of the loss function w.r.t. parameters β_j is

   $$\nabla L = (\sum_{i=1}^{n}(y_i - \hat{y}_i)\cdot y_i\cdot(1-y_i), \sum_{i=1}^{n}(y_i - \hat{y}_i)\cdot y_i\cdot(1-y_i)\cdot x_{i1}, \ldots, \sum_{i=1}^{n}(y_i - \hat{y}_i)\cdot y_i\cdot(1-y_i)\cdot x_{ik})$$

   where $y_i = \sigma(f(\vec{x}_i))$ is the output of the perceptron and $\hat{y}_i$ are the class labels in the training data.
   Hint: Apply the chain rule to calculate partial derivatives $\frac{\partial L}{\partial\beta_j}$ and use $\sigma'(x) = \sigma(x)(1-\sigma(x))$.

   (c) Consider a univariate regression $\hat{y} = wx$ where $w \in \mathbb{R}$, and $x \in \mathbb{R}^{1\times m}$. The cost function is the squared-error cost $\mathcal{J} = \frac{1}{m}||\hat{y} - y||^2$. Find $\frac{\partial\mathcal{J}}{\partial w}$ by performing a step-by-step procedure. 3P

2. **Activation functions and Neural Networks**

   (a) Implement the following functions that take a matrix as an input: 2P

   a) Softmax,
   b) Sigmoid,
   c) ReLu,
   d) Tanh.

   Hint: use numpy package functions np.sum, np.exp, np.maximum.

   (b) Use the two moons dataset. Split the dataset to 80% train data and 20% test data. Implement the following models: 2P

   - Logistic Regression using the default parameters from sklearn.
   - Neural Network using Pytorch:

  - Create 2 hidden layers where the output dimension of the first layer is 6.
  - Use the activation function *torch.tanh* after the first layer and *torch.sigmoid* after the second layer.
  - Use *BCELoss* as a loss function.
  - Use *SGD* as an optimisation function, with learning rate 0.0001.
  - Train the model for 500 epochs.
- Same Neural Network model as above but change learning rate to 0.04.

Calculate the accuracy score of all the models for the test set. Explain the role of learning rate in neural networks and explain the difference in accuracy scores between the two Neural Network models. Plot the decision boundary using the Neural Network with learning rate 0.04 for the test set.