Statistical Machine Learning: Exercise 4

Neural Networks, Support Vector Machines and Gaussian Processes

Prof. Dr. Kristian Kersting, Karl Stelzner, Claas Voelcker

TECHNISCHE UNIVERSITÄT DARMSTADT

Summer Term 2020

Due date: 2020/08/02, 23:59 PM

Task 1: Neural Networks [30+[15]=30 Points + 15 Bonus]

In this exercise, you will use the dataset `mnist_small`, divided into four files. The *mnist* dataset is widely used as benchmark for classification algorithms. It contains 28x28 images of handwritten digits (pairs `<input, output>` correspond to `<pixels, digit>`).

1a) Multi-layer Perceptron (30 Points)

Implement a neural network and train it using backpropagation on the provided dataset. Choose your loss and activation functions and your hyperparameters (number of layers, neurons, learning rate, ...), briefly explaining your choices. You **cannot** use any library beside `numpy`! That is, you have to implement by yourself the loss and activation functions, the backpropagation algorithm and the gradient descent optimizer (if you want to use any). Experiment around with hyperparameters and check what works and what does not.

Show how the misclassification error (in percentage) on the testing set evolves during the learning. An acceptable solution achieves an error of 8% or less. Attach snippets of your code.

Hint: If your network does not learn, check how the network parameters change and plot the trend of your loss function. You will get full points, if you include a thorough discuss of the observed problems.

1b) [Bonus] Deep Learning (15 Points)

In recent years, deep neural networks have become one of the most used tools in machine learning. Highlight the qualitative differences between classical neural networks and deep networks. Which limitations of classical NN does deep learning overcome? Give an intuition of the innovations introduced in deep learning compared to traditional NN. Are deeper networks necessarily better from a theoretical perspective? (Hint: Have a look at this paper. Use Google Scholar to read other scientific papers for more insights.)

Task 2: Support Vector Machines [3+2+2+8+4+6+[10]=25 Points + 10 Bonus]

In this exercise, you will use the dataset `iris-pca.txt`. It is the same dataset used for Homework 3, but the data has been pre-processed with PCA and only two kinds of flower ('Setosa' and 'Virginica') have been kept, along with their two principal components. Each row contains a sample while the last attribute is the label (0 means that the sample comes from a 'Setosa' plant, 2 from 'Virginica').
You are allowed to `numpy` functions (e.g., `numpy.linalg.eig`). For quadratic programming we suggest `cvxopt`.

2a) Definition (3 Points)

Briefly define SVMs. What is their advantage w.r.t. other linear approaches we discussed this semester?

2b) Quadratic Programming (2 Points)

Formalize SVMs as a constrained optimization problem.

2c) Slack Variables (2 Points)

Explain the concept behind slack variables and reformulate the optimization problem accordingly.

2d) Slack Variables (8 Points)

Solve the optimization problem of the previous question. Show all the intermediate steps and write down the final solution.

2e) The Dual Problem (4 Points)

What are the advantages of solving the dual instead of the primal?

2f) Kernel Trick (6 Points)

Explain the kernel trick and why it is particularly convenient in SVMs.

2g) [Bonus] Implementation (10 Points)

Learn an SVM to classify the data in `iris-pca.txt`. Choose your kernel and decide whether to use slack variables or not. Create a plot showing the data, the support vectors and the decision boundary. Show also the misclassified samples. Attach a snippet of your code.

Task 3: Gaussian Processes [10 Bonus]

3a) [Bonus] GP Regression (10 Points)

Implement a Gaussian Process to fit the target function $y = \sin(x) + \sin^2(x)$ with $x \in [0, 0.005, 0.01, 0.015, \ldots, 2\pi]$. Use a squared exponential kernel, an initial mean of 0 and assume a noise variance of 0.001. Begin with no target data points and, at each iteration, sample a new point from the target function according to the uncertainty of your GP (that is, sample the point where the uncertainty is the highest) and update it. Plot your GP (mean and two times standard deviation) after iterations 1, 2, 4, 8 and 16. In each figure, plot also the true function as ground truth and add a new marker for each new sampled point. Attach a snippet of your code.