Mathematics of machine learning (winter term 2024/25)

Exercise sheet I

October 29th, 2024

1. Task. (Implementing the Perceptron algorithm)
Implement the perceptron algorithm in Python yourself. You can use the prepared function template `my_perceptron.py` or in `perceptron_script.ipynb` provided on the OPAL course page and complete the marked lines.

2. Task. (Fooling the Perceptron – Part I)
Consider the Perceptron and linear hypotheses $\mathcal{L}_d$. We want to calculate a perturbation $\lambda \mathbf{v} \in \mathbb{R}^d$ with $\lambda \in \mathbb{R}$ and $\mathbf{v} \in \mathbb{R}^d$ such that for a correctly classified object with feature $\mathbf{x} \in \mathbb{R}^d$ and label $y \in \{-1, +1\}$, i.e

$$h_{\mathbf{w},b}(\mathbf{x}) = y \iff y(\mathbf{w} \cdot \mathbf{x} + b) > 0$$

the corresponding perturbed feature $\mathbf{x} + \lambda \mathbf{v}$ is classified incorrectly, i.e

$$h_{\mathbf{w},b}(\mathbf{x} + \lambda \mathbf{v}) \neq y \iff y(\mathbf{w} \cdot (\mathbf{x} + \lambda \mathbf{v}) + b) < 0.$$

a) Suppose the direction $\mathbf{v} \in \mathbb{R}^d$ is given. Depending on $y \in \{-1, +1\}$ determine a corresponding scaling λ such that $y(\mathbf{w} \cdot (\mathbf{x} + \lambda \mathbf{v}) + b) < 0$.

b) In the case of linear hypotheses $h_{\mathbf{w},b}$ which perturbation direction $\mathbf{v}$ do you think is most promising?

c) Suppose the features $\mathbf{x}$ must always be nonnegative, because, e.g. they describe the grayscale values in black and white images. How would you modify (depending on $y \in \{-1, +1\}$ the direction $\mathbf{v}$ from subtask b) so that $\mathbf{x} + \lambda \mathbf{v}$ also has only nonnegative entries?

d) Apply these results to fooling the Perceptron for the MNIST data set. Use the provided Jupyter notebook `perceptron_script_MNIST.ipynb`

3. Task. (Fooling the Perceptron – Part II)
We consider again the Perceptron and linear hypotheses

$$h_{\mathbf{w},b}(\mathbf{x}) = \operatorname{sgn}(\mathbf{w} \cdot \mathbf{x} + b), \qquad \mathbf{x} \in \mathbb{R}^d.$$

This time, we want to compute the *closest* misclassified perturbation $\widetilde{\mathbf{x}} \in \mathbb{R}^d$ of a correctly classified object with feature vector $\mathbf{x}$, i.e.,

$$h_{\mathbf{w},b}(\mathbf{x}) = y \qquad \text{but} \qquad h_{\mathbf{w},b}(\widetilde{\mathbf{x}}) \neq y.$$

Here we use the Euclidean distance $\|\cdot\|$ in $\mathbb{R}^d$ to measure the distance between feature vectors $\mathbf{x}, \widetilde{\mathbf{x}} \in \mathbb{R}^d$.

a) Derive a constrained optimization problem describing this task.

b) Implement and solve this constrained optimization problem in Python using `scipy optimization` routines such as `minimize` and apply it to the MNIST data from the previous task. Again use the provided Jupyter notebook `perceptron_script_MNIST.ipynb`.
Compare the result (and computation time) to the approach of the previous task (Part I).

4. Task. (Training the Perceptron)
We now take a closer look at running the Perceptron algorithm for the MNIST data set.

a) Split the MNIST data of "7" and "8"s into a training subset and a test subset. Use the routine `train_test_split` for that purpose. Use only 70% of the given data for training, the other part for testing.

b) Run the Perceptron algorithm for the training data for $T = 5000$ iterations, save all the iterates $\mathbf{w}'_t$, $t = 1, \ldots, T$ and the associated empirical risks.

Then, compute for all iterates $\mathbf{w}'_t$ also their *test error* or *test risk*, i.e.,

$$\frac{1}{m_{\text{test}}} \sum_{i=1}^{m} \ell\left(h_{\mathbf{w}_t, b_t}, (\mathbf{x}_i, y_i)\right)$$

where the summation is over all data pairs $(\mathbf{x}_i, y_i)$ in the test subset and ℓ is chosen again as 0-1-loss.

c) Plot the corresponding empirical risk (training error) and test risk versus the iteration t. Use a log scale for the y-axis. What do you notice?

d) When can we stop the training of the Perceptron? And what is the generalization error of the learned hypothesis?

Homework:
Compute Novikoff's runtime bound R^2B^2 for the Perceptron algorithm for the MNIST data set in Python. Why don't you need to run the Perceptron algorithm anymore once you computed the bound?