Lab Class ML:VI

Until Wednesday, Jan. 4th, 2024, 11:59 pm CET, solutions to the following exercises must be submitted as one zip-file named `ml23-ex4-group<your-group-number>.zip` via Moodle:
1, 2, 3, 4, 6a,b, and 7.

Exercise 1 : Gradient Descent (1+1=2 Points)

(a) Name one difference between the perceptron training rule and the gradient descent method.

(b) Name the difference in the algorithm between batch gradient descent and incremental gradient descent.

Exercise 2 : Perceptron Learning (2 Points)

Given two perceptrons $y_0()$ and $y_1()$ defined by *heaviside*$(\sum_{j=0}^{p} w_j x_j)$ as usual with identical weights except that $w_0 = 0$ for $y_0()$ but $w_0 = 1$ for $y_1()$. Is one of $y_0()$ and $y_1()$ *more general* than the other, and if yes, which one (or both)? Explain your answer.

Exercise 3 : Perceptron Learning (1+1+2+2+1=6 Points)

In this exercise, you design a single perceptron with two inputs x_1 and x_2. This perceptron shall implement the boolean formula $A \wedge \neg B$ with a suitable function $y(x_1, x_2)$. Use the values 0 for *false* and 1 for *true*.

(a) Draw all possible examples and a suitable decision boundary in a coordinate system.

(b) Draw the graph of the perceptron. The schematic must include x_1, x_2, and all model weights.

(c) Manually determine a set of suitable weights $\mathbf{w} = (w_0, w_1, w_2)$ from your drawings.

(d) Now determine $\mathbf{w}$ using the perceptron training algorithm (PT). Use a learning rate η of 0.3 and initialize the weights with $w_0 = -0.5$ and $w_1 = w_2 = 0.5$. Instead of selecting examples randomly, use the following examples in the given order (stop after those four examples):

x_1	x_2	c
0	0	0
0	1	0
1	0	1
1	1	0

Draw the decision boundary after every weight update into the coordinate system of (a).

(e) Briefly describe one effect of changing the learning rate η on the learning progress.

Exercise 4 : Perceptron Learning (1 Points)

Why can the boolean formula A *XOR* B not be learned by a single perceptron? Justify your answer with a drawing.

Exercise 5 : Multilayer Perceptron

Consider the minimum multilayer perceptron that can handle the XOR problem (slide ML:IV-59), but with an activation function $\mathbf{f}(\mathbf{z}) = (f(z_1), \ldots, f(z_d))^T$ instead of *Heaviside*(). This is a two-layer perceptron with $p = 2$ attributes, a hidden layer with $l = 2$ units, and an output layer with $k = 1$ units.
Therefore, as per slide ML:IV-82:

$$\mathbf{y}(\mathbf{x}) = \mathbf{f}\left(W^{\mathsf{o}}\begin{pmatrix}1\\ \mathbf{f}(W^{\mathsf{h}}\mathbf{x})\end{pmatrix}\right) = \mathbf{f}\left((w^{\mathsf{o}}_{10}, w^{\mathsf{o}}_{11}, w^{\mathsf{o}}_{12})\begin{pmatrix}1\\ \mathbf{f}\left(\begin{pmatrix} w^{\mathsf{h}}_{10} & w^{\mathsf{h}}_{11} & w^{\mathsf{h}}_{12}\\ w^{\mathsf{h}}_{20} & w^{\mathsf{h}}_{21} & w^{\mathsf{h}}_{22}\end{pmatrix}\begin{pmatrix}1\\ x_1\\ x_2\end{pmatrix}\right)\end{pmatrix}\right)$$

Confirm for yourself that thus

$$\mathbf{y}(\mathbf{x}) = f(w^{\mathsf{o}}_{10} + w^{\mathsf{o}}_{11} \cdot f(w^{\mathsf{h}}_{10} + w^{\mathsf{h}}_{11} \cdot x_1 + w^{\mathsf{h}}_{12} \cdot x_2) + w^{\mathsf{o}}_{12} \cdot f(w^{\mathsf{h}}_{20} + w^{\mathsf{h}}_{21} \cdot x_1 + w^{\mathsf{h}}_{22} \cdot x_2))$$

Show that this network is not able to learn a solution to the XOR problem if the scalar activation function $f()$ is the identity function: $f(z) = z$.

Exercise 6 : Parameters of the Multilayer Perceptrons (2+1+0+0=3 Points)

In this exercise, you analyze the number of weights (parameters) of multilayer perceptrons. We use the notation from the lecture (e.g., slide ML:IV-104), where multilayer perceptrons have d layers, p attributes, hidden layer i with l_i units, and an output layer with k units.

(a) Let $d = 4$, $p = 7$, $l_1 = 5$, $l_2 = 3$, $l_3 = 3$, and $k = 4$. Draw the graph of the multilayer perceptron.

(b) Calculate the number of weights in the multilayer perceptron of (a).

(c) Calculate the number of weights in the multilayer perceptron of (a) but with each l_i doubled, i.e., $l_1 = 10$, $l_2 = 6$, $l_3 = 6$. Has the number of weights doubled as well?

(d) Let $f(p, l_1, \ldots, l_{d-1}, k)$ be a function that computes the number of weights in the general case. Write down an expression for f.

Exercise 7 : P Multi-Class Classification with Neural Networks (1+1+1+1+1+1=6 Points)

In this exercise, you will implement a two-layer perceptron for predicting binary argument quality.

Download and use these files from Moodle (the `tsv` files are the same as in the last sheet):

- `features-train-cleaned.tsv`: Feature vectors for each example in the training set.
- `features-test-cleaned.tsv`: Feature vectors for each example in the test set.
- `quality-scores-train-cleaned.tsv`: Quality scores for each example in the training set.
- `programming_exercise_neural_networks.py`: Template for the programming exercise. It contains function stubs for each function mentioned below. The template contains code from our solution from the last exercise sheet that we can re-use. The program should be used like this with the files above:

```
python3 programming_exercise_neural_networks.py
  features-train-cleaned.tsv quality-scores-train-cleaned.tsv
features-test-cleaned.tsv quality-scores-test-predicted.tsv
```

- `requirements.txt`: Requirements file for the template; can be used to install dependencies.

(a) Implement a function `encode_class_values` to encode class values as vectors $\mathbf{c} \in \{0, 1\}^k$ (see slide ML:IV-100).

(b) Implement a function `predict_probabilities` that, given W_{h} and W_{o}, predicts the class probabilities for each example of a dataset.

Hint: You can find it in the code linked on the lecture slides.

(c) Implement a function `predict` that, given W_{h} and W_{o}, predicts the class for each example of a dataset (as the one with the highest probability).

(d) Implement a function `train_multilayer_perceptron` that fits a multilayer perceptron using the IGD Algorithm (slide ML:IV-100). Like in the last exercise sheet, make sure to return the misclassification rate on both training and validation set for plotting, but this time also return the weights after each iteration (for task (e)).

Hint: You can find it in the code linked on the lecture slides.

(e) Select the model (iteration) that achieved the best misclassification rate on the validation set and use it to predict the overall quality for each example in the test set (`features-test-cleaned.tsv`). Write the prediction as one column to a file and submit that along with your other solutions.

(f) Which lines of your code would you need to change from two classes to all three? Mark each of them with the comment "`# change for 3 classes`".

If you would like to improve your model, here are some hints of what you could try:

- Try experimenting with adjusting various hyperparameters of the model (e.g. learning rate, number of hidden units, etc).
- Try to decrease the learning rate with each iteration.
- Try to implement one of the feature scaling methods explained in this video by Andrew Ng.
- Implement k-fold cross-validation instead of using a single hold-out in order to get a more robust estimate of your model's performance on the unseen data.
- Expand your implementation to support perceptrons with more than two layers. Hint: the parameter `l` can be replaced by a list of layer sizes; the weight matrices for the hidden layers can also be stored in a list.