Deep Learning

Exercise 13: Exam Preparation

Room: **BIN-1-B.01**

Instructor: Manuel Günther

Email: guenther@ifi.uzh.ch

Office: AND 2.54

Friday, June 3, 2022

Outline

1. Review
2. Exam Preparation

Outline

1 Review

Review of the Exercise

1. Python Programming

- Basic introduction to `python`, `numpy`, `matplotlib`
  $\rightarrow$ No plotting in exam!
- Perceptron learning rule

2. Gradient Descent

- Given: loss function $\mathcal{J}_{\vec{w}}$
- Compute gradient manually
- Iterative gradient descent
- Linear regression

3. Non-Linear Regression

- Two-layer network
- Gradient implementation
- Iterative gradient descent
- Function approximation

4. Multi-Output Networks

- Stochastic gradient descent
- Implement batch
- Implementation of network via `numpy` matrices

Review of the Exercise

5. Classification

- Binary cross-entropy loss
- Binary classification network
- SoftMax implementation
- Categorical cross-entropy loss
- Classification accuracy

PyTorch

- Building a network
  - → Different layers
  - → Activation functions
- `torchvision` datasets
- Input transforms
- `DataLoader`'s
- Loss functions
- Optimizers
- Basic training loop

Review of the Exercise

6. Convolutional Networks

- `Sequential` networks
- Convolutional layers
- Pooling and striding
- Flattening

7. Transfer Learning

- Overwrite functions in ResNet
- Extract deep features
- Nearest neighbor classification

8. Open-Set Learning

- Dataset split and filtering
- Adapted SoftMax loss
- Autograd function

9. Auto-Encoder Network

- Encoder-decoder networks
- Fract.-strided convolution
- Image reconstruction from manipulated deep features

Parenthesis

Correction of Autograd Function

- Autograd `Function` does not compute the *Jacobian*
- It computes the *gradient* of the function
- Example derivative chain (two-layer fully-connected network):

$$\frac{\partial \mathcal{J}}{\partial \mathbf{W}^{(1)}} = \frac{\partial \mathcal{J}}{\partial \mathbf{Z}} \frac{\partial \mathbf{Z}}{\partial \mathbf{H}} \frac{\partial \mathbf{H}}{\partial \mathbf{A}} \frac{\partial \mathbf{A}}{\partial \mathbf{W}^{(1)}} \qquad \frac{\partial \mathcal{J}}{\partial \mathbf{W}^{(2)}} = \frac{\partial \mathcal{J}}{\partial \mathbf{Z}} \frac{\partial \mathbf{Z}}{\partial \mathbf{W}^{(2)}}$$

- `backward` gets the gradient w.r.t. the output as parameters
  $\rightarrow$ $\frac{\partial \mathcal{J}}{\partial \mathbf{Z}}$ in the above example
- It returns the gradients with respect to its inputs: $\frac{\partial \mathcal{J}}{\partial \mathbf{H}}$ and $\frac{\partial \mathcal{J}}{\partial \mathbf{W}^{(2)}}$
  $\rightarrow$ Usually needs to multiply the Jacobian with the input gradient

Parenthesis

The backward Function

- Defined as static method via `@staticmethod` decorator
- Has two parameters: context `ctx`, gradient of output `grad`

```
@staticmethod
def backward(ctx, grad):
```

- Extract stored information from context

```
param1, param2 = ctx.saved_tensors
```

- Return gradient for **each input of** `forward`
  - → Need to be of same shape as input parameters; the applied Jacobian
  - → Can be `None` if derivative for one parameter makes no sense

```
derivative_for_param1 = grad * ...
return derivative_for_param1, None
```

Review of the Exercise

10. Recurrent Network

- Simple recurrent network
- Supervised training with unlabeled dataset
- Iterative text generation

11. Adversarial Training

- Backpropagation to input
- Create adversarial images
- Combined training schedule

12. RBF Networks

- Custom layer implementation
- Parameter initialization
- Custom activation function
- Batch implementation of RBF
- Bottleneck network plot

Outline

2 Exam Preparation

Exam Preparation

Examination

- Friday 17.6.2022, 8:00 - 10:00
- Online only, no proctoring
- Several assignments
- Several tasks per assignment
  → Theoretical and practical
- Jupyter notebook per assignment
  → Several tasks in one notebook
- Login 15 minutes before
- 2 hours time (might be short)
  → I might reduce required points

Examination Type

- Open-book exam
- Any resources allowed
  → Except for human resources

Test Run

- Participate in the test run
- Available next week
  → 7.6.2021 – 13.6.2021
- Test download and upload
- Installation instruction

Exam Preparation

EPIS Test Setup

- Free text boxes for theory
- Multiple-choice questions
- Single-choice question
  - → Only for practical tasks
  - → Not required to click (but required to finish)
- One file upload boxes
  - → Upload notebook of assignment (includes several tasks)

Typing Math in Text Boxes

- No symbols or LaTeX enabled
- Write simple text:
- Use `df/dx` to indicate $\frac{\partial f}{\partial x}$
- Use `a^b` to indicate a^b
- Use `a_b` to indicate a_b
- Use parentheses `()` `[]` `{}`
- Anything readable is accepted

Exam Preparation

Running of Experiments

- Template provided in Jupyter notebook
  - → Includes empty spaces . . . to be filled
  - → Code for automatic download of data
  - → Some additional code for other purposes
  - → No need to wait for training to finish
- Usage of Google Colaboratory encouraged: http://colab.research.google.com
  - → Make sure to be familiar with the environment
  - → Know how to upload and download notebooks (as `.ipynb`) from Colab
- Usage of local installation possible
  - → See installation instructions on next page

Exam Preparation

Local Installation

- Download and install Conda on your machine:
  http://docs.conda.io/projects/conda/en/latest/user-guide/install
- Open the conda prompt (`bash`, `cmd.exe`, Anaconda prompt, ...)
- Create new environment: `conda create -n DL python=3.8`
- Activate environment: `conda activate DL`
- Install required packages:
  ```
  conda install numpy scipy matplotlib jupyter
      pytorch torchvision cudatoolkit -c pytorch
  ```
- Test your installation: Jupyter notebook from test run

Exam Preparation

Running of Experiments Locally

- **Optionally** pre-download (encrypted) data (170 MB)
  http://seafile.ifi.uzh.ch/f/99ecdad37758492c8179/?dl=1
  → Decrypt to local directory: password during exam
  → Notebooks will automatically download, both locally or on Colab
- Download Jupyter notebooks to same directory from OLAT
- Run from console (will open browser): `jupyter notebook [file].ipynb`
  → Use `Ctrl-C` on console to stop when finished
- Maybe setup editor to work with Jupyter notebooks
  → For example: Visual Studio Code setup:
  https://code.visualstudio.com/docs/datascience/jupyter-notebooks