Hot Topics in Machine Learning Safety

Fundamentals

Seminar Schedule

- Introduction and Motivation
- ML Fundamentals 👈
- Testing
- Explainability
- Uncertainty Estimation
- Anomaly Detection
- Adversarial Machine Learning
- Alignment/Societal/Long-Term Risk

Agenda

- Computational Graphs
- Automatic Differentiation
- Optimization via Gradient Descent
- Regression
- Classification

Computational Graphs

- Different perspective on computation
- Foundation of modern Deep Learning frameworks

Computational Graphs

$$f(x, y) = \sqrt{\exp(x) + y}$$

We could implement this by:

- $a = \exp(x)$
- $b = a + y$
- $f = \sqrt{b}$

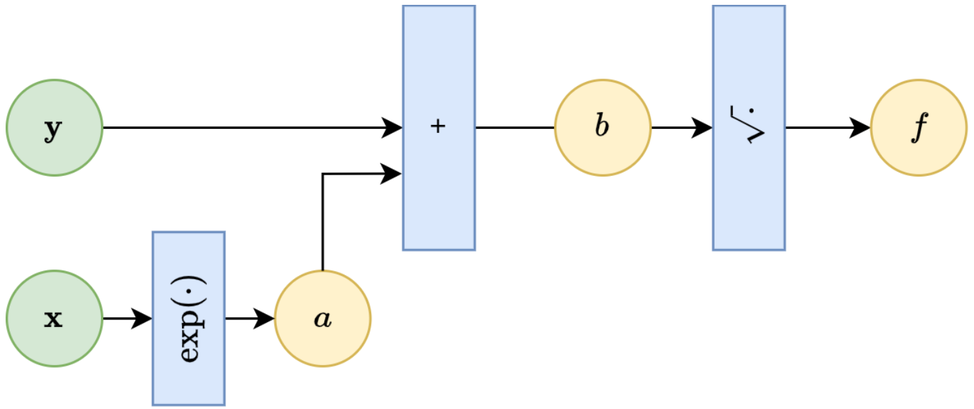
```python
from torch import tensor

x = tensor(1.0)
y = tensor(2.5)

a = x.exp()
b = a + y
f = b.sqrt()
```

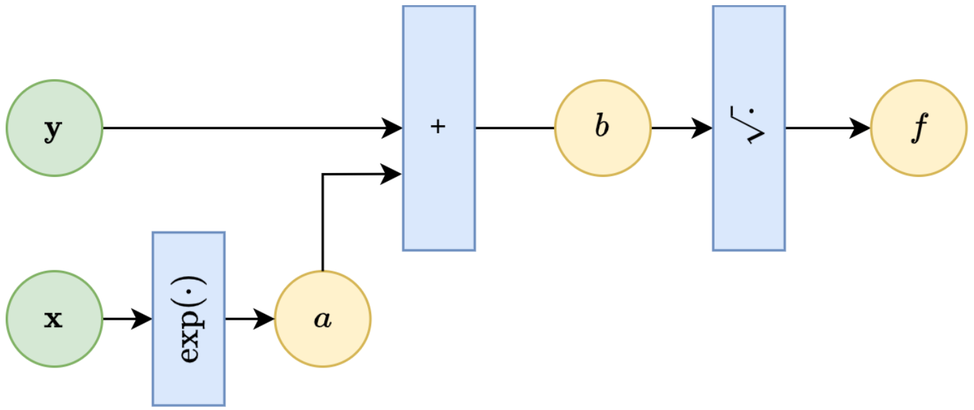
Computational Graphs

$$a = \exp(x) \quad b = a + y \quad f = \sqrt{b}$$



Automatic Differentiation

- I want to calculate derivatives, e.g. $\frac{\partial f}{\partial x_1}$, in the computational graph
- Algorithm: "Backpropagation"
- Repeated application of the chain rule: $\frac{\partial f}{\partial y} = \frac{\partial f}{\partial b} \cdot \frac{\partial b}{\partial y}$
- "Gradient": $\nabla_{\mathbf{x}} f = [\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}]^\top$



Automatic Differentiation

$$f(x, y) = \sqrt{\exp(x) + y}$$

$$\nabla_x f = [\frac{\partial f}{\partial x}]^\top \quad \nabla_y f = [\frac{\partial f}{\partial y}]^\top$$

```python
from torch import tensor

x = tensor(1.0, requires_grad=True)
y = tensor(2.5, requires_grad=True)

b = x.exp() + y
f = b.sqrt()
f.backward() # calculate gradients
print(x.grad) # tensor(0.5950)
print(y.grad) # tensor(0.2189)
```

Optimization via Gradient Descent

- We have a function $\mathcal{L}$
- We can calculate $\nabla \mathcal{L}$
- Given some initial x, we can iteratively tune x to minimize $\mathcal{L}(x)$

$$x^{(i+1)} = x^{(i)} - \alpha \nabla \mathcal{L}(x^{(i)})$$

High Learing Rate α

Low Learing Rate α

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Linear Regression

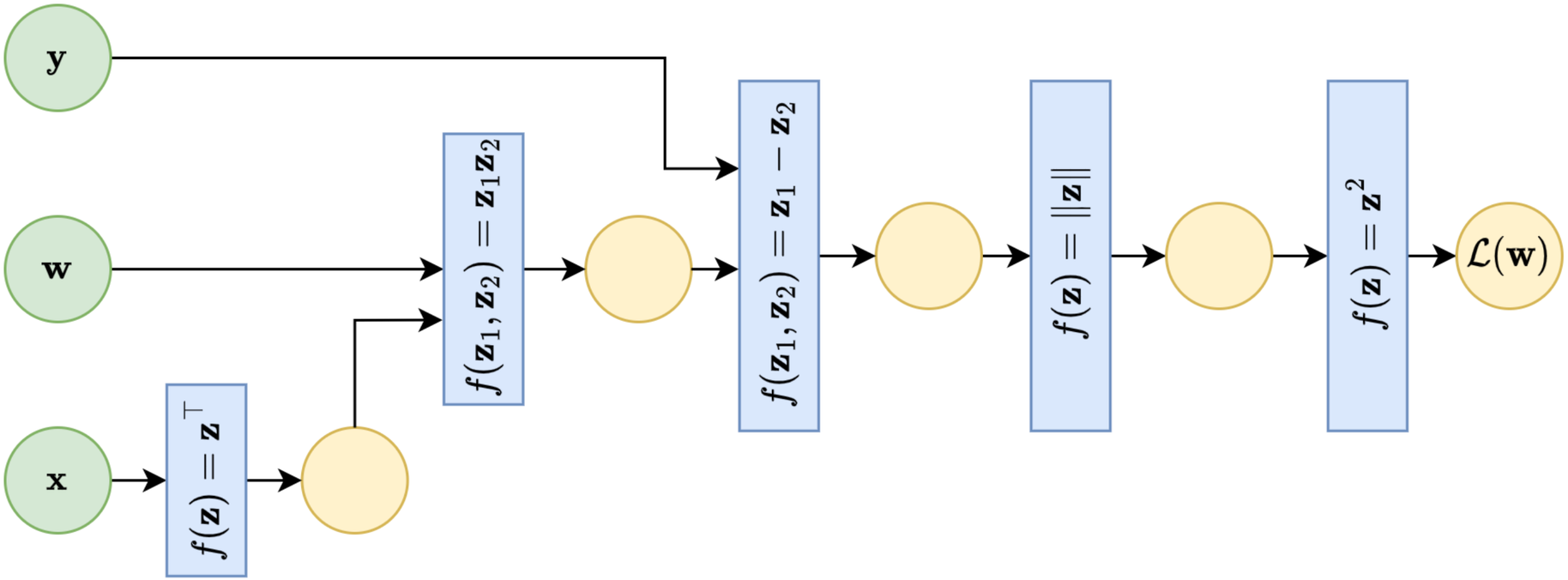
- Dataset $\mathcal{D} = \{x_i, y_i\}_{i=0}^{N}$
- Model $f_{\mathbf{w}} : \mathbb{R} \to \mathbb{R}$
- $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} = w_1 x_1$
- $\mathcal{L}(\mathbf{w}) = \|\mathbf{y} - f_{\mathbf{w}}(\mathbf{x})\|^2 = \|\mathbf{y} - \mathbf{x}^\top \mathbf{w}\|^2$

Optimize:

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} = \frac{1}{|\mathcal{D}|} \sum_{x,y \in \mathcal{D}} \mathcal{L}(\mathbf{w})$$

$$= \frac{1}{|\mathcal{D}|} \sum_{x,y \in \mathcal{D}} \|\mathbf{y} - f_{\mathbf{w}}(\mathbf{x})\|^2$$

Linear Regression



- We can use backprop to calculate $\frac{\partial \mathcal{L}}{\partial w_1}, \ldots, \frac{\partial \mathcal{L}}{\partial w_n}$
- We could equally calculate $\frac{\partial \mathcal{L}}{\partial x_1}, \frac{\partial \mathcal{L}}{\partial y_1}, \ldots$

Non-Linear Regression

- Dataset $\mathcal{D} = \{x_i, y_i\}_{i=0}^{N}$
- Deep Neural Network $f_\theta : \mathbb{R} \rightarrow \mathbb{R}$
- $\mathcal{L}(\mathbf{w}) = \|\mathbf{y} - f_\theta(\mathbf{x})\|^2$

Optimize:

$$\theta^\star = \arg\min_\theta = \frac{1}{|\mathcal{D}|} \sum_{x,y \in \mathcal{D}} \mathcal{L}(\theta)$$

$$= \frac{1}{|\mathcal{D}|} \sum_{x,y \in \mathcal{D}} \|\mathbf{y} - f_\theta(\mathbf{x})\|^2$$

Linear Classification

- Dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=0}^{N}$
- Model $f_{\mathbf{W}} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$
- $\mathbf{W} \in \mathbb{R}^{2\times 2}$
- Probability that x belongs to class y: $f_{\mathbf{W}}(\mathbf{x})_y$
- $f_{\mathbf{W}}(\mathbf{x})_y = \frac{\exp(\mathbf{W}_y^\top \mathbf{x})}{\sum_j \exp(\mathbf{W}_j^\top \mathbf{x})}$
- $\mathcal{L}(\mathbf{W}) = -\log(f_{\mathbf{W}}(\mathbf{x})_y)$

Optimize: average loss over dataset

High Learning Rate α

Regularization

Add penalty term to loss:

$$\mathcal{L}(\mathbf{W}) = -\log(f_{\mathbf{W}}(\mathbf{x})_y) + \lambda \|\mathbf{W}\|^2$$

Home Assignments (Optional)

- Download "Mathematics for Machine Learning" (Free)
  - https://mml-book.github.io/book/mml-book.pdf

- Read/Skim Chapter 5: "Vector Calculus" ($\approx$ 30 pages)
  - Section 5.6: Backpropagation and Automatic Differentiation

- E-Learning: PyTorch introduction Notebook
  - Jupyter Notebook with exercises, available soon