Natural Language Processing

Training LLMs

Marco Kuhlmann
Department of Computer and Information Science

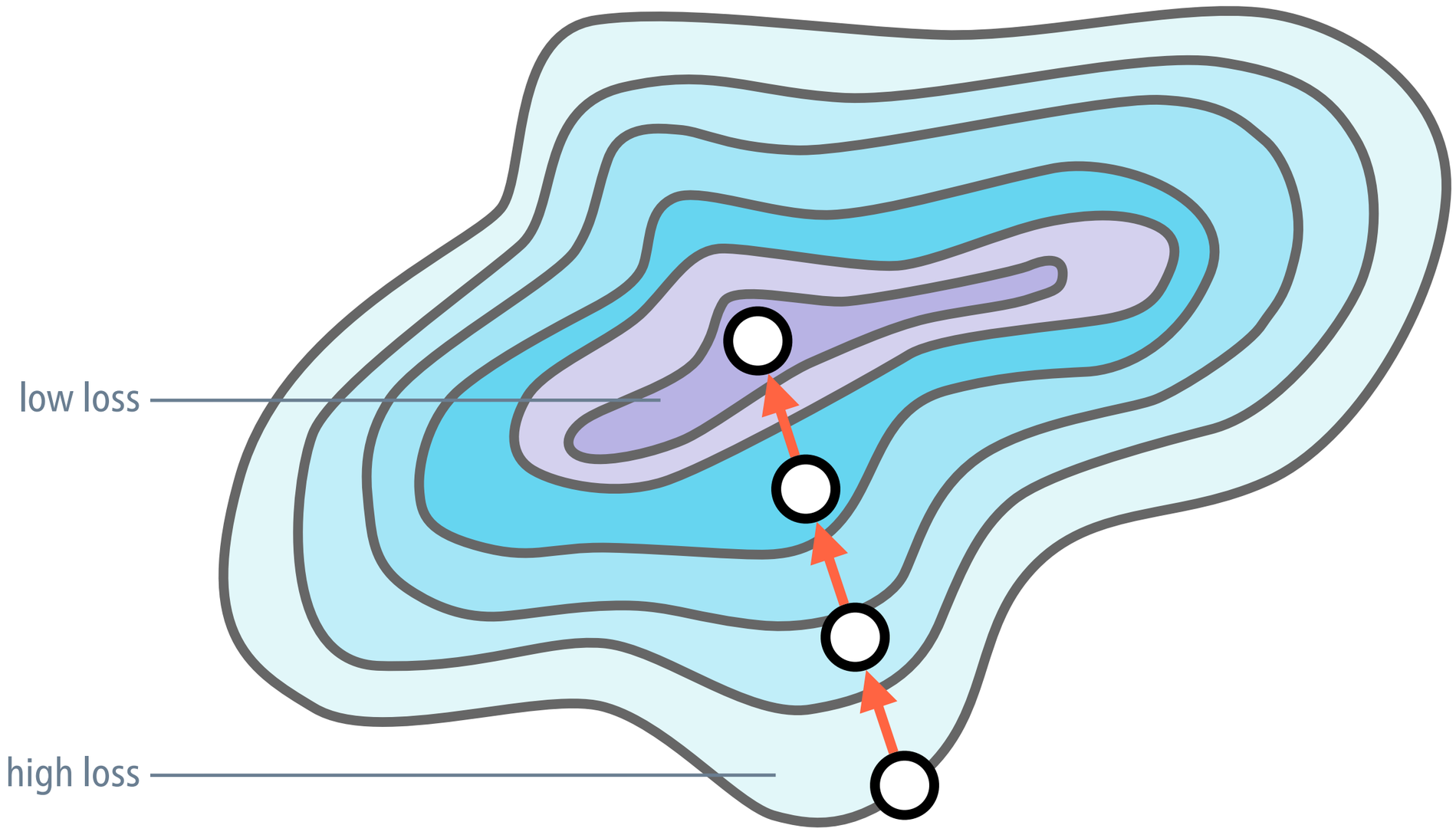
LINKÖPING UNIVERSITY

Gradient descent

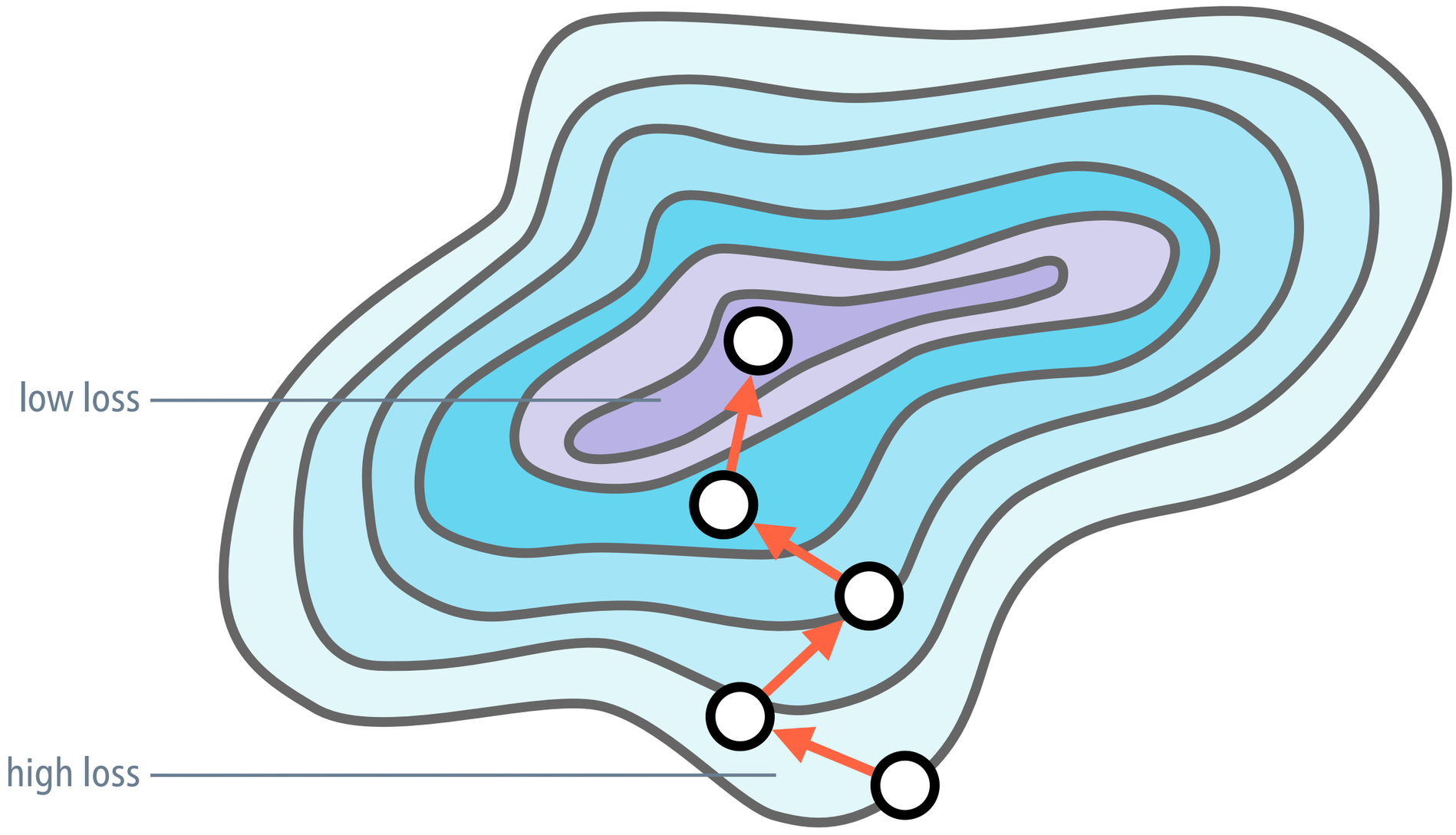
- **Step 0:** Start with random values for the parameters $\boldsymbol{\theta}$.
- **Step 1:** Compute the gradient of the loss function for the current parameter settings, $\nabla L(\boldsymbol{\theta})$.
- **Step 2:** Update the parameters $\boldsymbol{\theta}$ as follows: $\boldsymbol{\theta} \coloneqq \boldsymbol{\theta} - \alpha \nabla L(\boldsymbol{\theta})$

  The hyperparameter α is the learning rate.
- Repeat step 1–2 until the loss is sufficiently low.

Gradient descent



Stochastic gradient descent



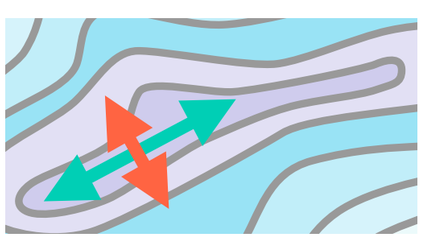
Unstable training

Training deep neural networks is often unstable in the initial phase:

- Parameters are initialised randomly, which means they are far from the optimal solution.
- Gradients are computed using relatively small subsets of the data, which causes a large variability between gradients.

Adam optimiser

- **Adam (Adaptive Moment Estimation)** is the most popular optimisation algorithm for training language models.
- Adam smoothes out gradient estimates by averaging past gradient directions and magnitudes.
- It maintains different learning rates per parameter, which helps it adapt to different regions of the optimisation surface.



large steps along the valley, small steps across the valley

Gradient clipping

- Excessively large gradients can cause gradient explosion and training instability.

- **Gradient clipping** stabilises the training process by downscaling gradients if they exceed a certain limit.

- Specifically, gradient clipping will rescale gradients if their total norm exceeds the specified threshold.

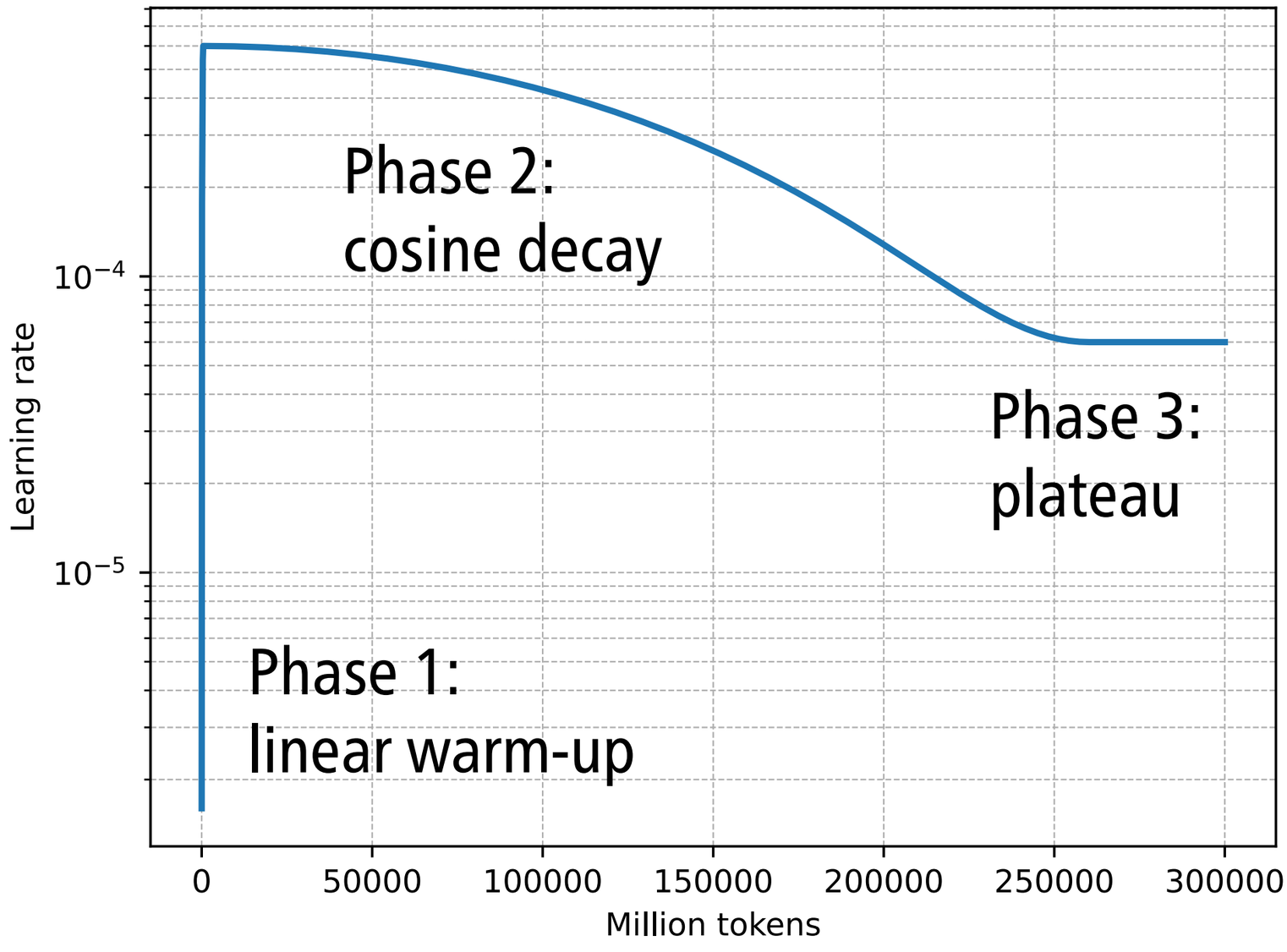
$$\text{scaling factor} = \frac{\text{max\_norm}}{\max(\text{total norm}, \text{max\_norm})}$$

Learning rate scheduling

- Choosing the right learning rate is crucial, but different training stages may require different learning rates.
- A **learning rate scheduler** is a strategy that adjusts the learning rate during training.

  change after a fixed number of steps; exponential decay; cyclic regime …

Learning rate scheduling



Gradient accumulation

- Larger batch sizes yield better estimates of the true gradient of the loss function but require more memory.

- **Gradient accumulation** breaks up the gradient computation across several smaller chunks.

- We compute the gradient for each micro-batch, add them up, and then do a single weight update with the accumulated gradient.

Gradient accumulation

```
optimizer.zero_grad()
n_elements = 0
for microstep in range(n_microsteps):
   x, y = next(data_loader)
   loss = F.cross_entropy(model(x), y, reduction="sum")
   loss.backward()
   n_elements += len(x)
loss = loss / n_elements
optimizer.step()
```

default is “mean”, which divides
by the number of elements in each mico-batch

Weight decay

- Large weights can lead to overfitting and prevent generalisation.
- **Weight decay** is a regularisation technique that penalises large weights by adding a scaled L2 norm of the weights to the loss:

$$L_{\text{reg}}(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|^2$$

- It is common to not weight-decay biases and other one-dimensional tensors, such as those in layer norms.