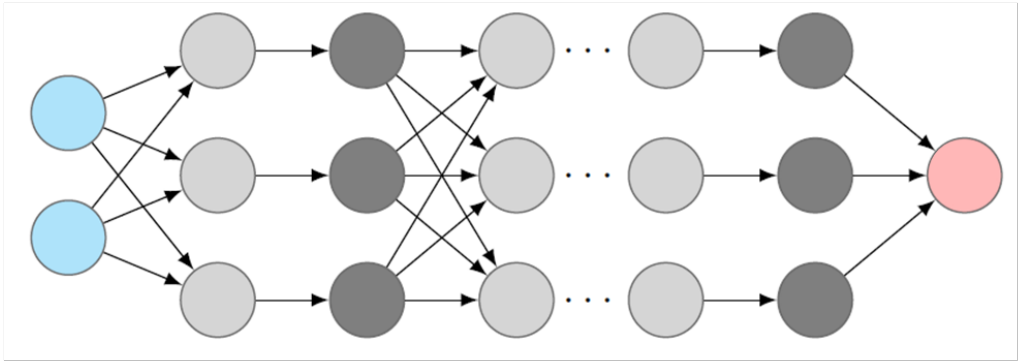
Training fully connected networks

- Stochastic Gradient Descent -

Lecturer: Michael Möller – michael.moeller@uni-siegen.de
Exercises: Hartmut Bauermeister – hartmut.bauermeister@uni-siegen.de



Gradient descent (GD)

We understood: For a continuously differentiable $E : \mathbb{R}^n \to \mathbb{R}$, the quantity $-\nabla E(\theta)$ points into the direction of steepest descent. GD moves into this direction!

$$\theta(k+1) = \theta(k) - \tau \nabla E(\theta(k))$$

New parameters	Previous parameters	Direction of steepest descent

Is this cheap or expensive? That depends on E!

Common situation: *Can easily consist of 1,000,000 summands!*

$$E(\theta) = \sum_{\text{training examples } j} \mathcal{L}(\mathcal{N}(x_j;\theta), y_j)$$

$$\mathcal{N}(x;\theta) = \ell^L(\ell^{L-1}(\ldots(\ell^1(x;\theta^1)\ldots);\theta^{L-1});\theta^L)$$

Stochastic gradient descent (SGD)

Idea for:

$$E(\theta) = \sum_{\text{training examples } j} \mathcal{L}(\mathcal{N}(x_j; \theta), y_j)$$

Use only a few summands to compute an approximate gradient:

$$E_k(\theta) = \sum_{j \in I(k)} \mathcal{L}(\mathcal{N}(x_j, \theta), y_j)$$ for a ***very small index set*** $I(k)$

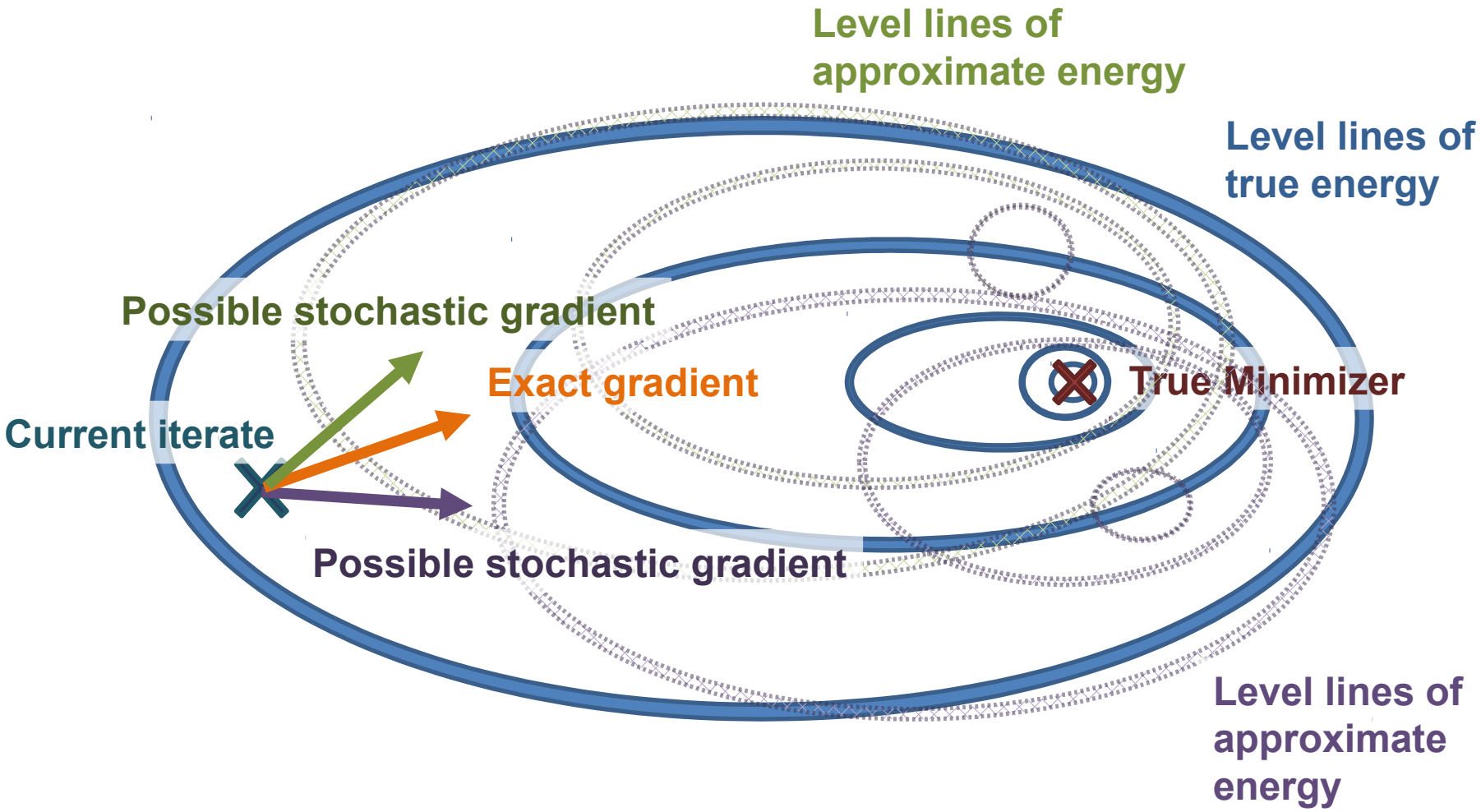
Update the parameters using this approximation

$$\theta(k+1) = \theta(k) - \tau \nabla E_k(\theta(k)) \approx \theta^k - \tau \nabla E(\theta^k)$$

Randomly selecting entries in the index set $I(k)$ leads to the name ***stochastic gradient descent***. The training examples (x_j, y_j) with $j \in I(k)$ are called a ***mini-batch***.

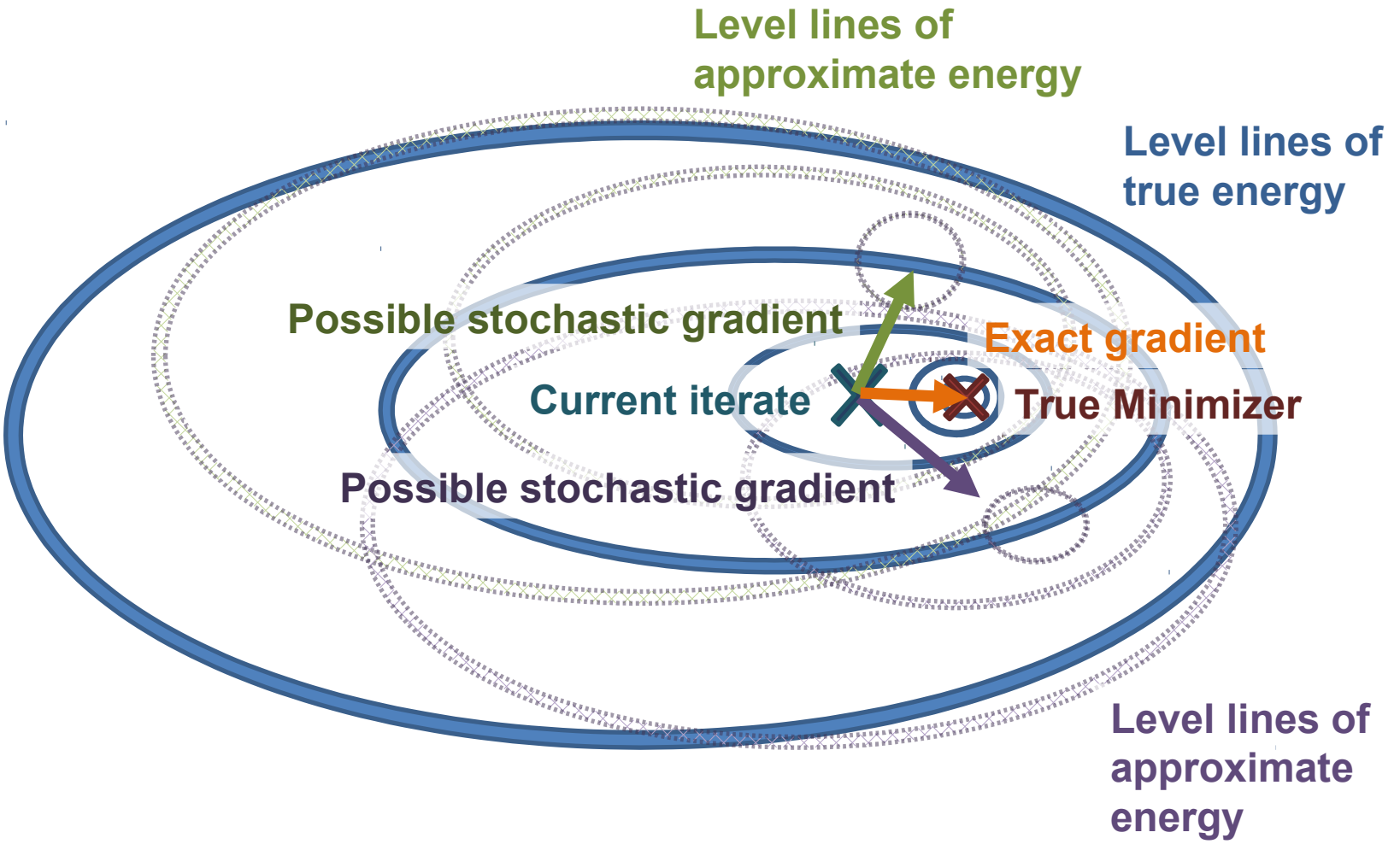
Stochastic gradient descent (SGD)

Approximating the gradients work well if one is still ``sufficiently far away'' from the true minimizer



Stochastic gradient descent (SGD)

Approximating the gradients can easily fail if we are ``sufficiently close'' to the true minimizer



Stochastic gradient descent (SGD)

Sanity check to be discussed in the lecture for a simple problem like

$$E(\theta) = \frac{1}{2}(\theta - 1)^2 + \frac{1}{2}(\theta + 1)^2$$

1. Once we are “close to the minimizer” (e.g. between 1 and -1 above), the approximate gradient might not point into the right direction anymore.
2. We need a stepsize that converges to zero, i.e., we need

   $$\theta(k+1) = \theta(k) - \tau(k)\ \nabla E_k(\theta(k)) \qquad \text{with} \qquad \lim_{k\to\infty} \tau(k) = 0$$

   to have a chance to converge.
3. The convergence speed at which $\tau(k)$ goes to zero may not be too fast,

   e.g. $\tau(k) = \frac{1}{2^k}$ fails!

Stochastic gradient descent (SGD)

Theoretical result: In order for

$$\lim_{k\to\infty} \mathbb{E}[\|\nabla E(\theta(k))\|^2] = 0$$

the step size $\tau(k)$ needs to satisfy

$$\sum_{k=1}^{\infty} \tau(k) = \infty, \qquad \sum_{k=1}^{\infty} \tau(k)^2 < \infty$$

Popular choice of stepsize: ***AdaGrad***

$$c(k+1) = c(k) + \|\nabla E_k(\theta(k))\|^2, \qquad \tau(k) = (c(k+1))^{-1/2}$$

$$\theta(k+1) = \theta(k) - \tau(k)\ \nabla E_k(\theta(k))$$

A thorough convergence analysis was published as a preprint e.g. in https://arxiv.org/pdf/1806.01811.pdf in June 2018.

Stochastic gradient descent (SGD)

Practical implementation of SGD for deep learning:

- **For** a desired number of *epochs* (= outer iterations)
  - Shuffle your training data!
  - **For** i **in** range (0 , total_number_of_training_examples , minibatch_size)
    - Take the chunk i:i+minibatch_size out of the (shuffeled) training data
    - Do a gradient descent step with a suitable step size (e.g. AdaGrad) only using the sum of loss functions on the current chunk (called *mini-batch*)

As we will learn in the next lecture, the plain SGD step in the above update can often be improved using a technique called *momentum*, or further accelerations such as in an algorithm called *adam*.