

Auto-Differentiation

- Back-propagation and the chain rule
- From single-layer to multi-layer perceptrons
- Autodifferentiation

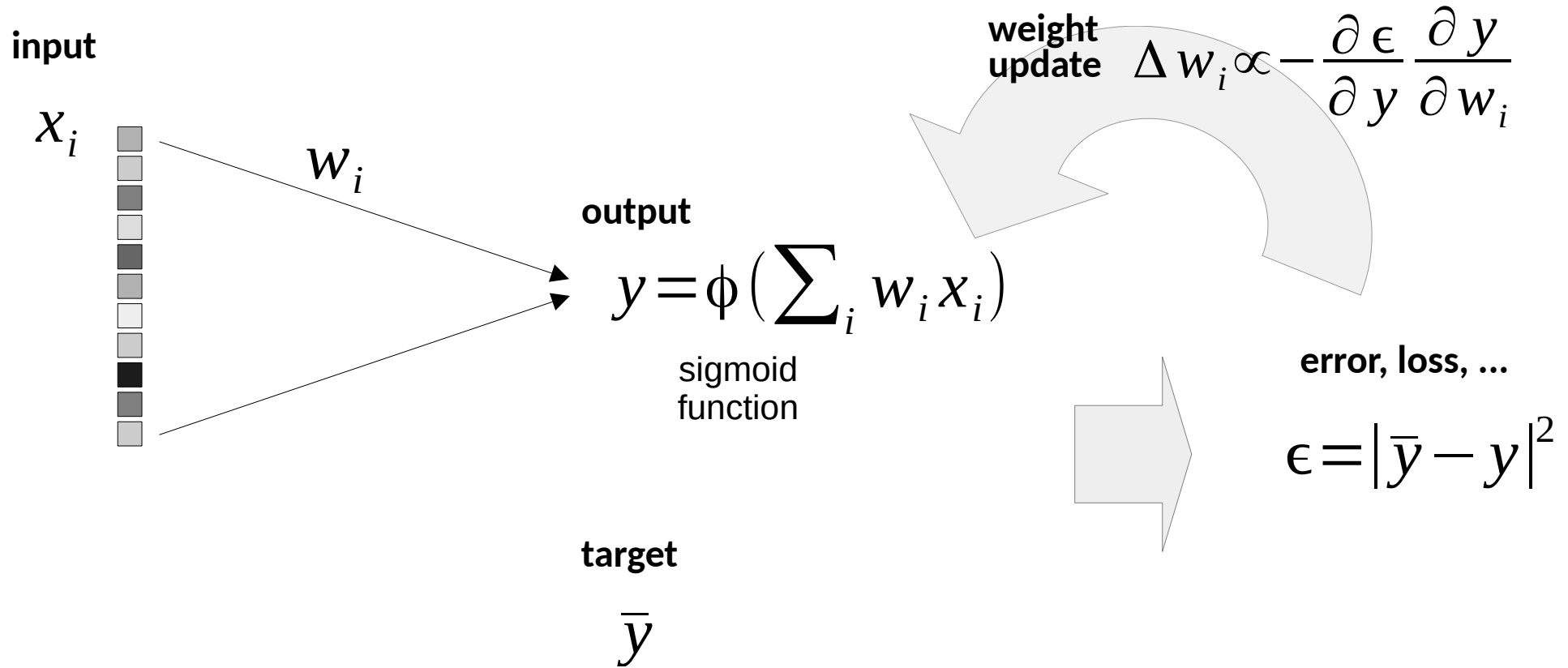
Prerequisites and References

- Supervised Learning:
 - Bishop (2006) *Machine learning and Pattern Recognition*;
https://www.cs.uoi.gr/~arly/courses/ml/tmp/Bishop_book.pdf
 - scikit-learn documentation: <https://scikit-learn.org/>
- Pytorch:
 - <https://pytorch.org/docs/stable/index.html>
 - <https://pytorch.org/tutorials/>

Auto-Differentiation

- Back-propagation and the chain rule
- From single-layer to multi-layer perceptrons
- Autodifferentiation

Logistic Regression (a.k.a. Nonlinear Perceptron)



Logistic Regression (a.k.a. Nonlinear Perceptron)

input

x_i



w_i

output

$$y = \phi \left(\sum_i w_i x_i \right)$$

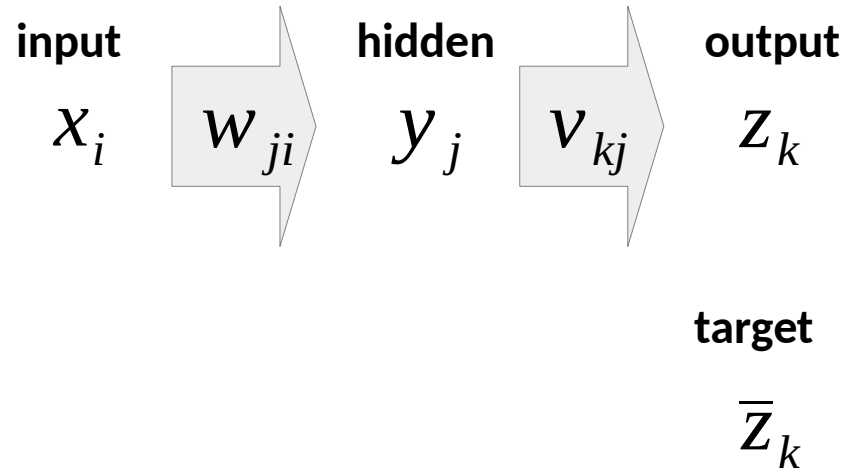
sigmoid
function

target

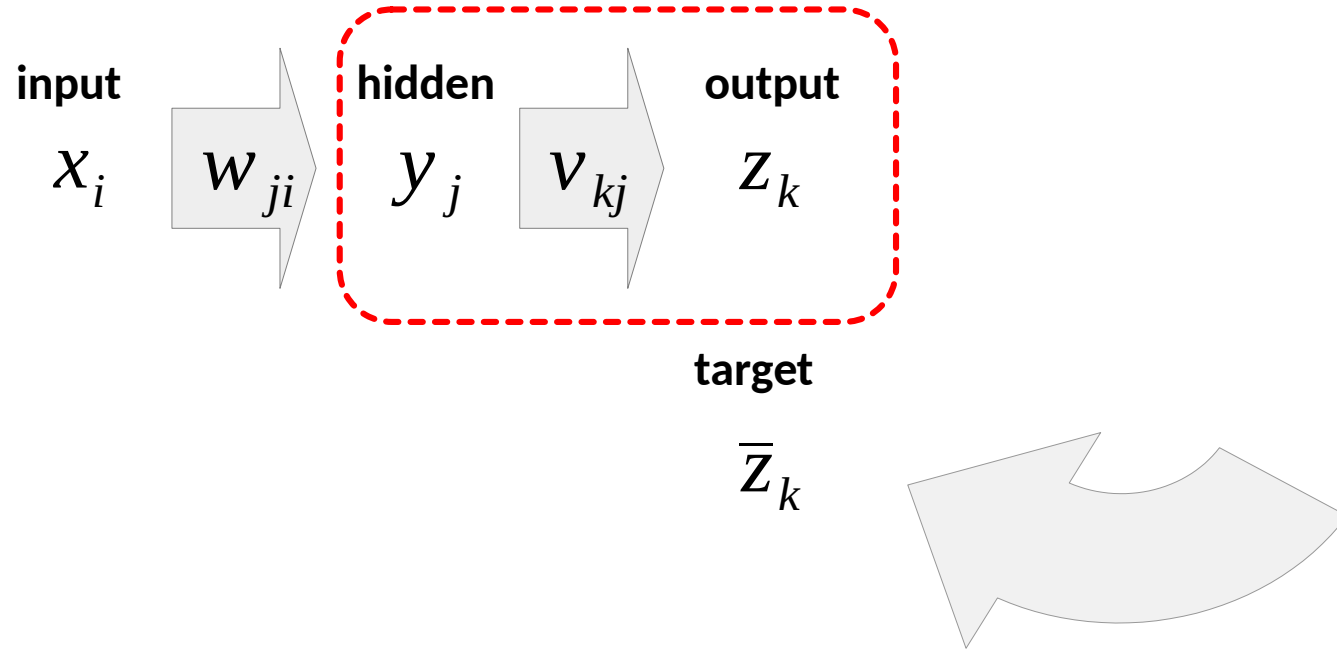
\bar{y}

- gradient descent
- depends on loss function and activation function ϕ
- extend to other network architecture?

Two-Layer Perceptron



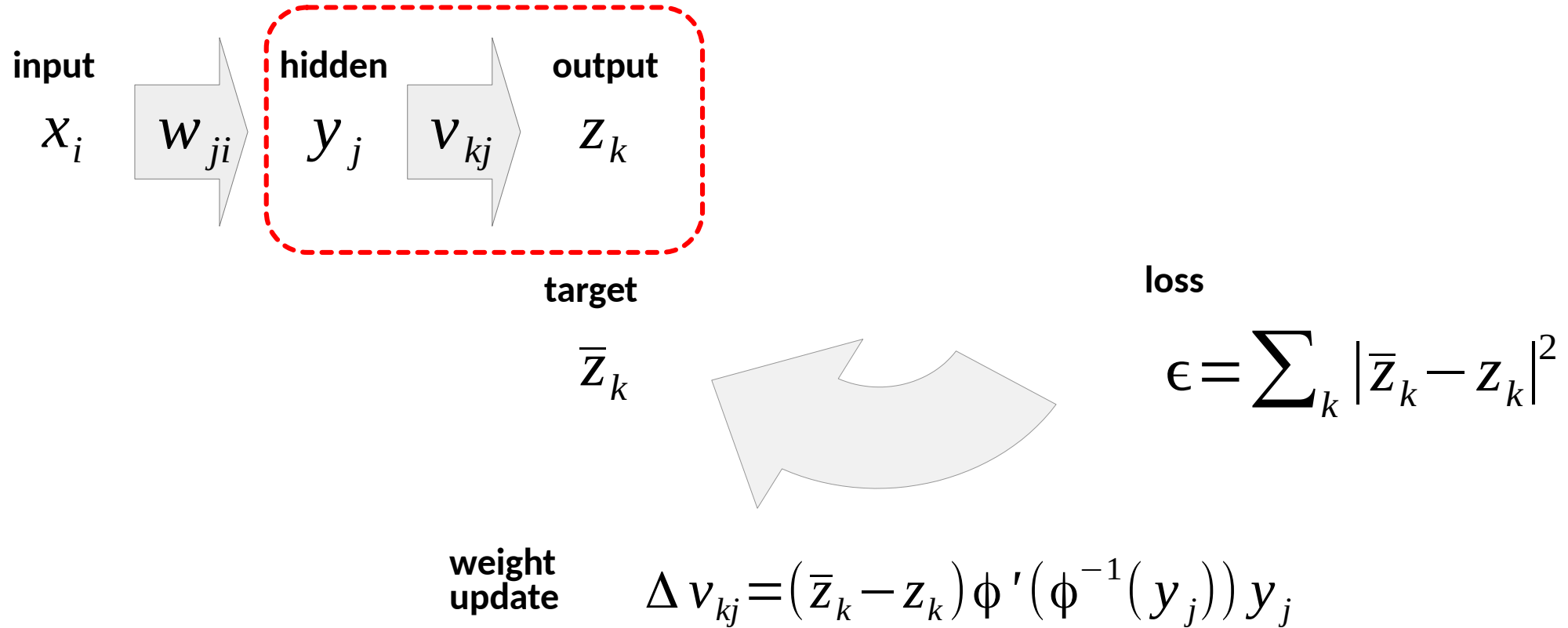
Two-Layer Perceptron



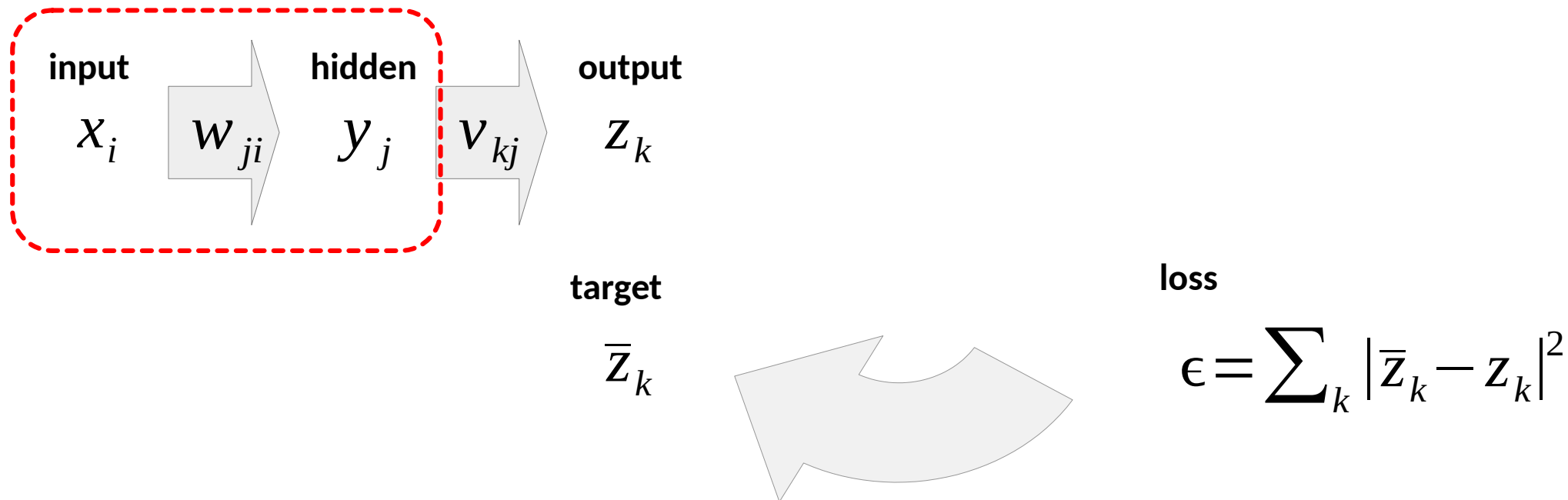
loss

$$\epsilon = \sum_k |\bar{z}_k - z_k|^2$$

Two-Layer Perceptron



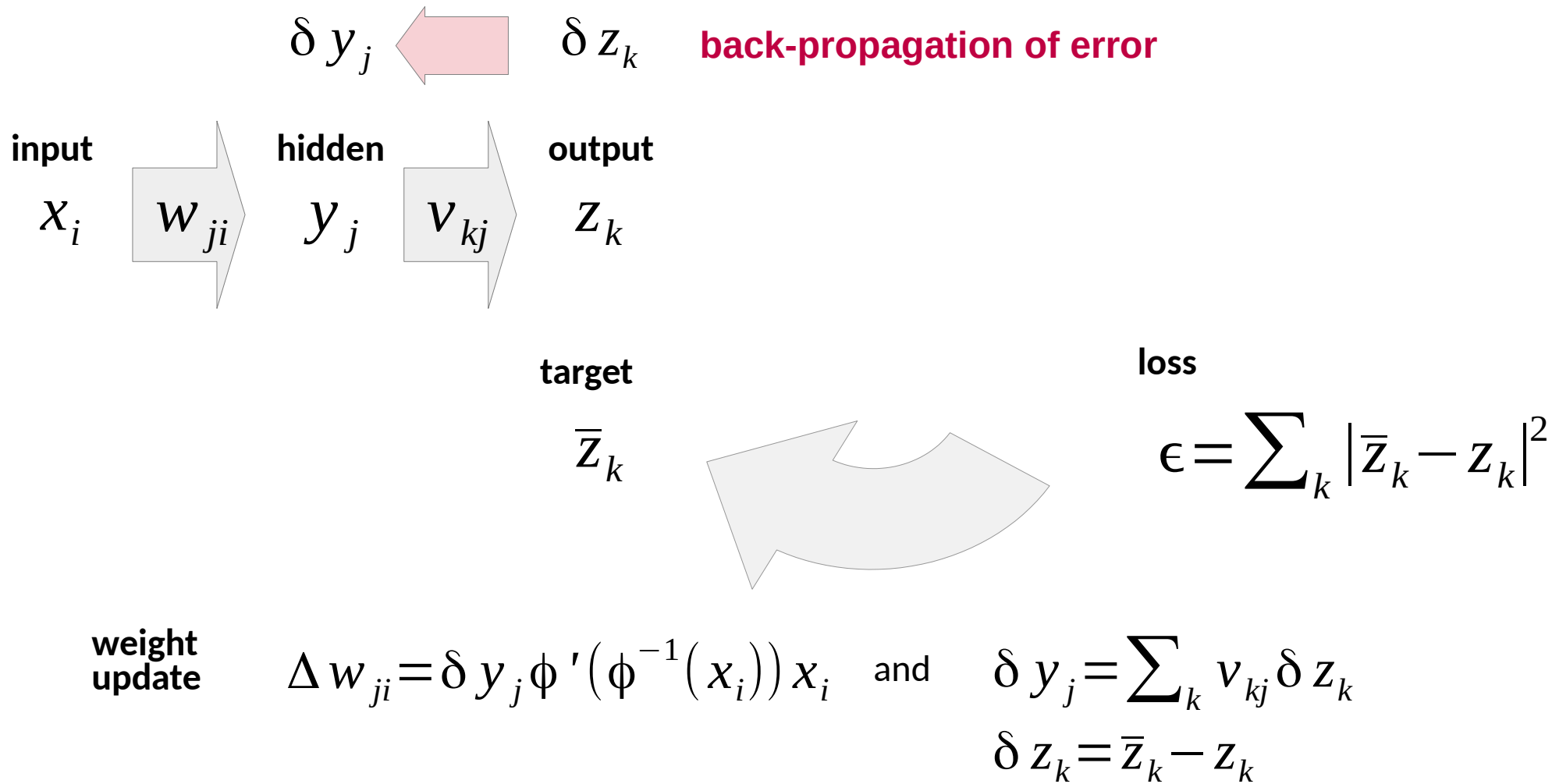
Two-Layer Perceptron



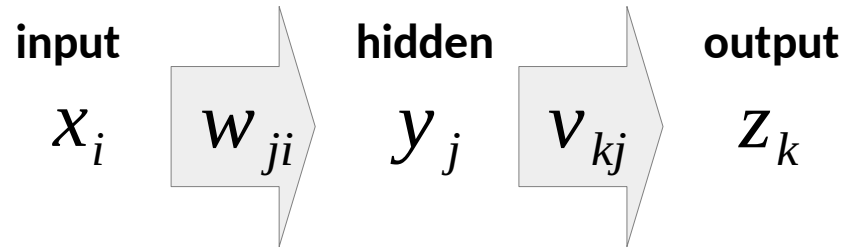
**weight
update**

$$\Delta w_{ji} = \delta y_j \phi'(\phi^{-1}(x_i)) x_i \quad \text{and} \quad \delta y_j = \sum_k v_{kj} \delta z_k$$
$$\delta z_k = \bar{z}_k - z_k$$

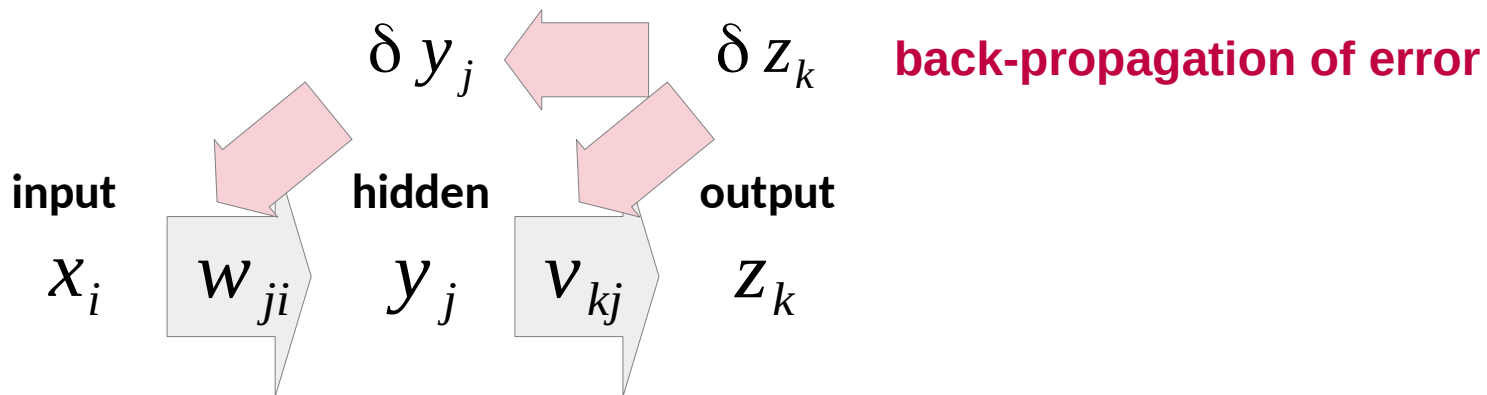
Two-Layer Perceptron



Autodifferentiation



Autodifferentiation



- just need to define forward model (calculations)
- automatic calculation of parameter updates (weights, etc.)



Practice

- Tutorials pytorch
- Tutorials JAX
- Multilayer perceptron, autoencoder, convolutional network