

Machine Learning Course



Part 1: Models in Computational Neuroscience

Part 2: Vision (L Perrinet)

Part 3: Supervised Learning

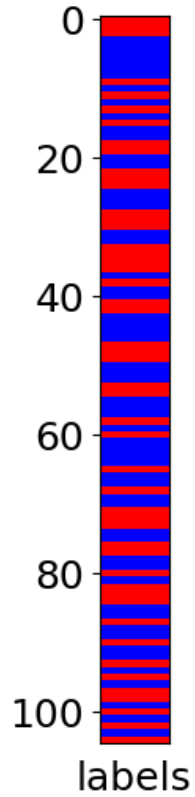
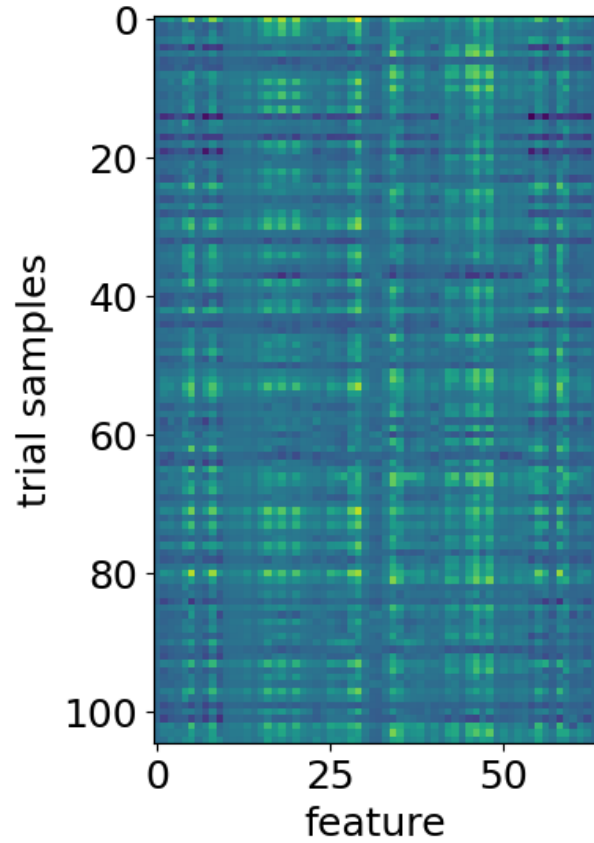
Part 4: Unsupervised Learning

Part 5: Time Series Classification

Part 3: Supervised Learning

- Basics
 - Cross-validation
 - Classifiers: basics
 - Hyperparameter tuning with nested cross-validation
 - Feature selection (RFE)
- Application to real datasets
- Training of logistic regression; autodifferentiation

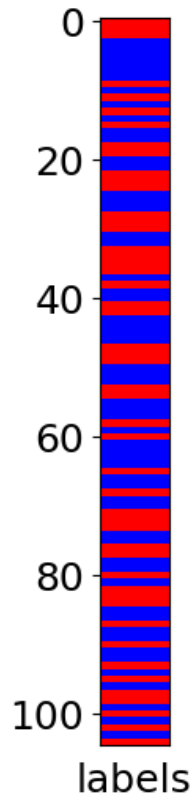
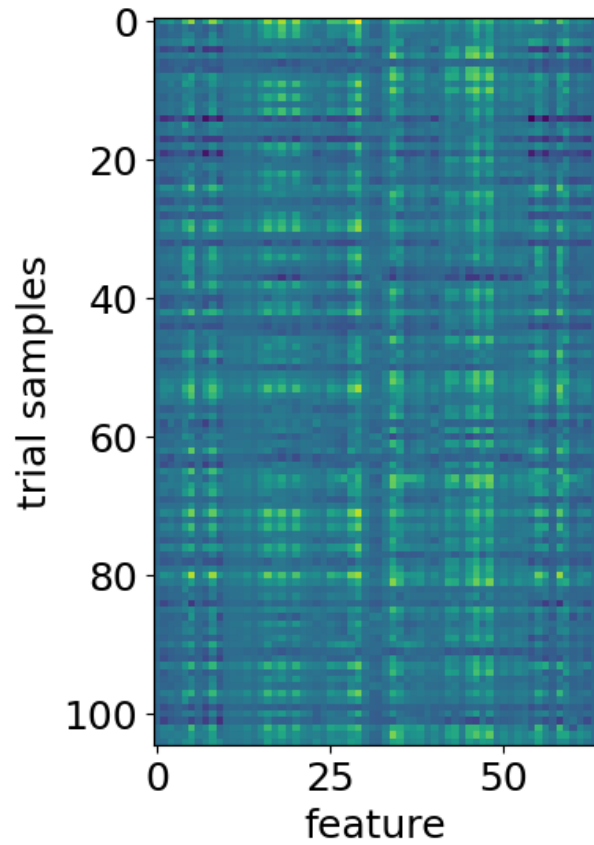
Cross-Validation



whole
dataset



Cross-Validation

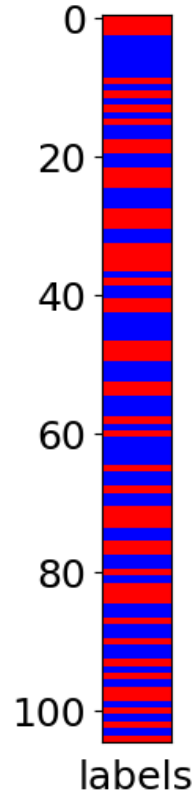
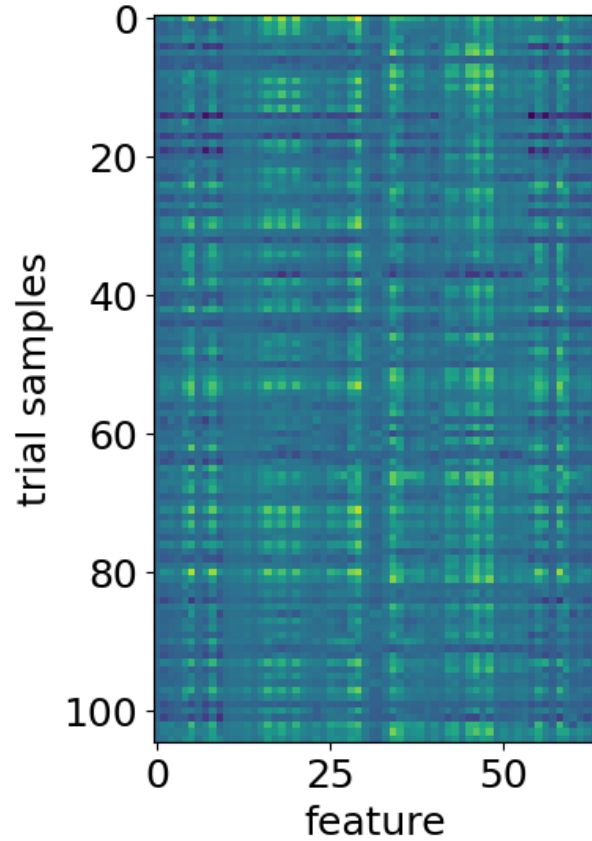


test set (20%)

train set (80%)



Cross-Validation



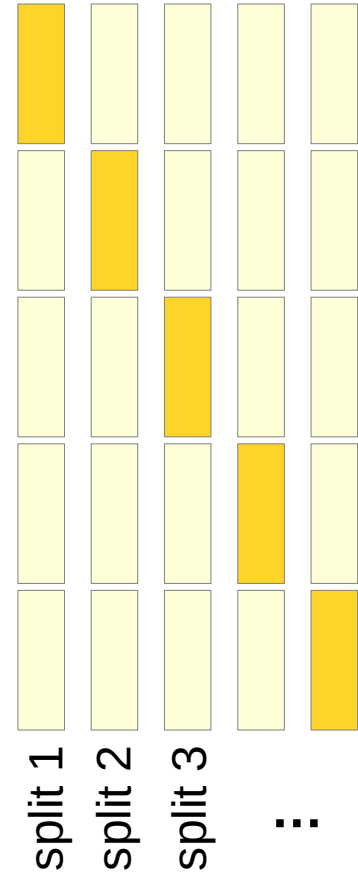
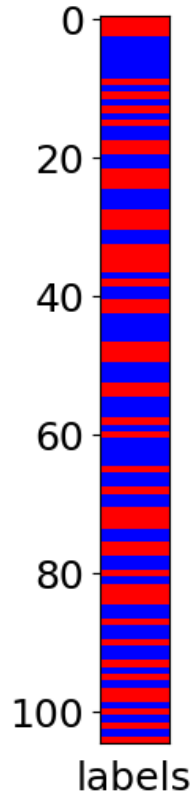
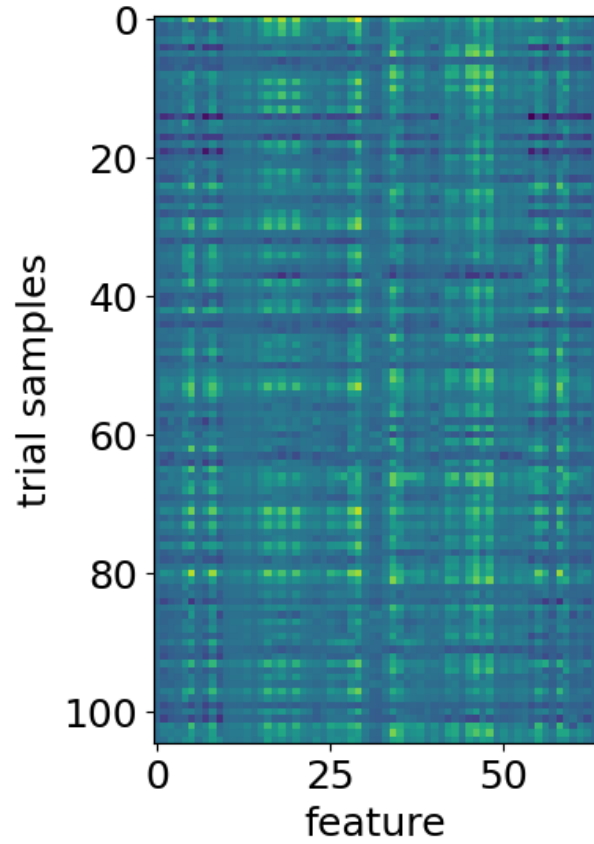
test set (20%)

train set (80%)

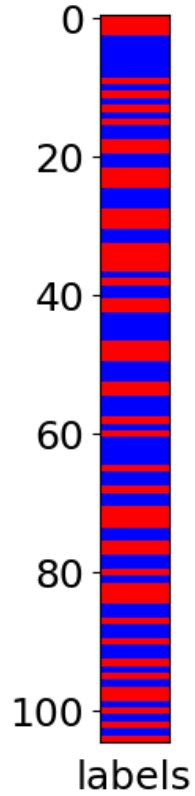
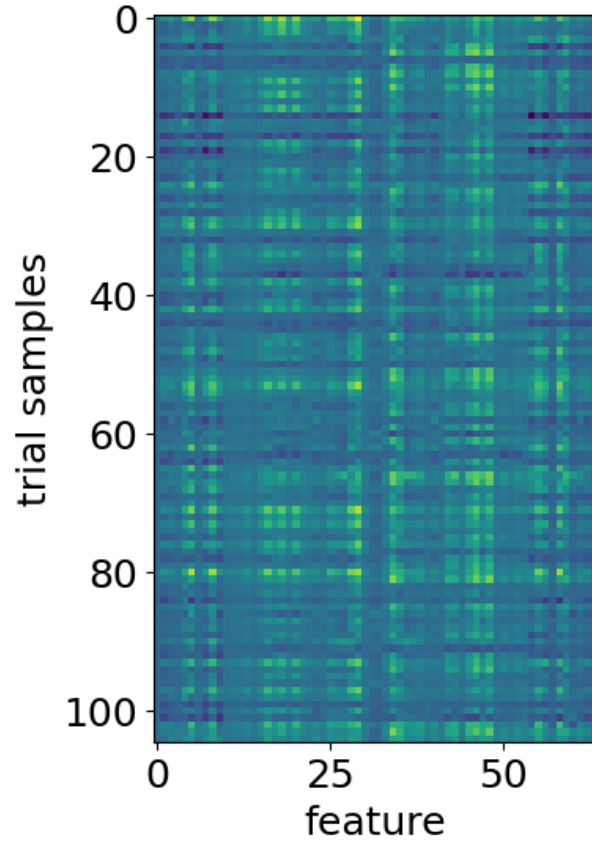


test accuracy:
generalizability
to “new”
(unseen) data

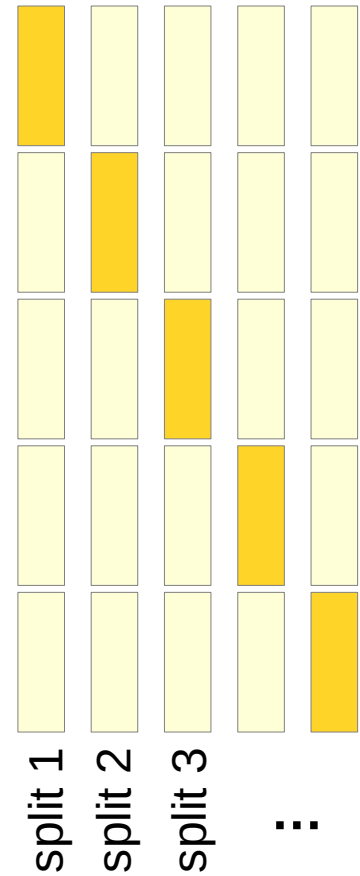
Cross-Validation



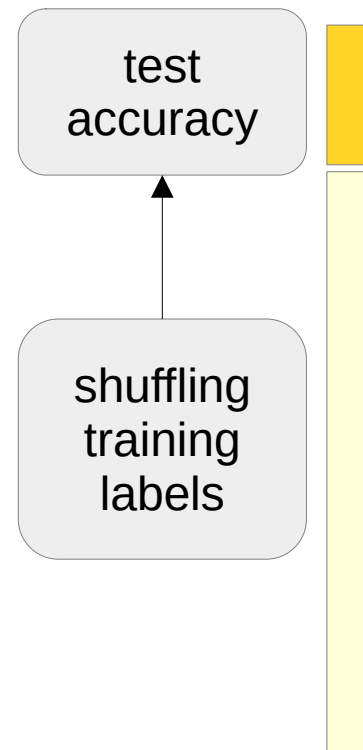
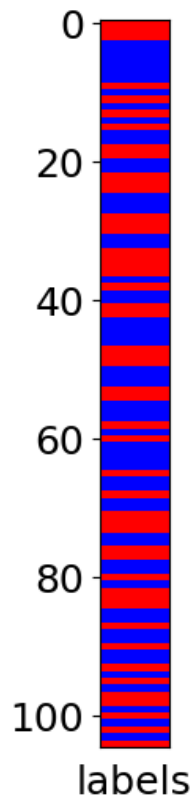
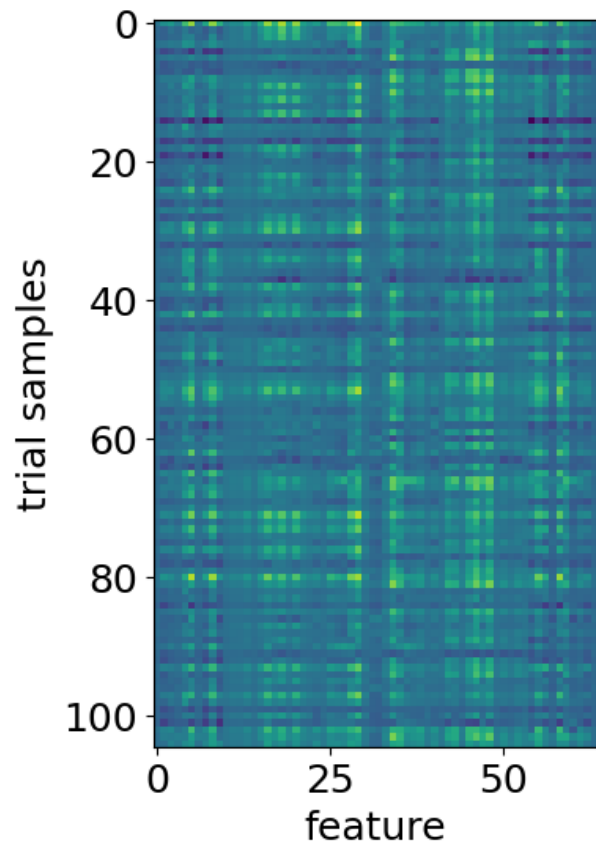
Cross-Validation



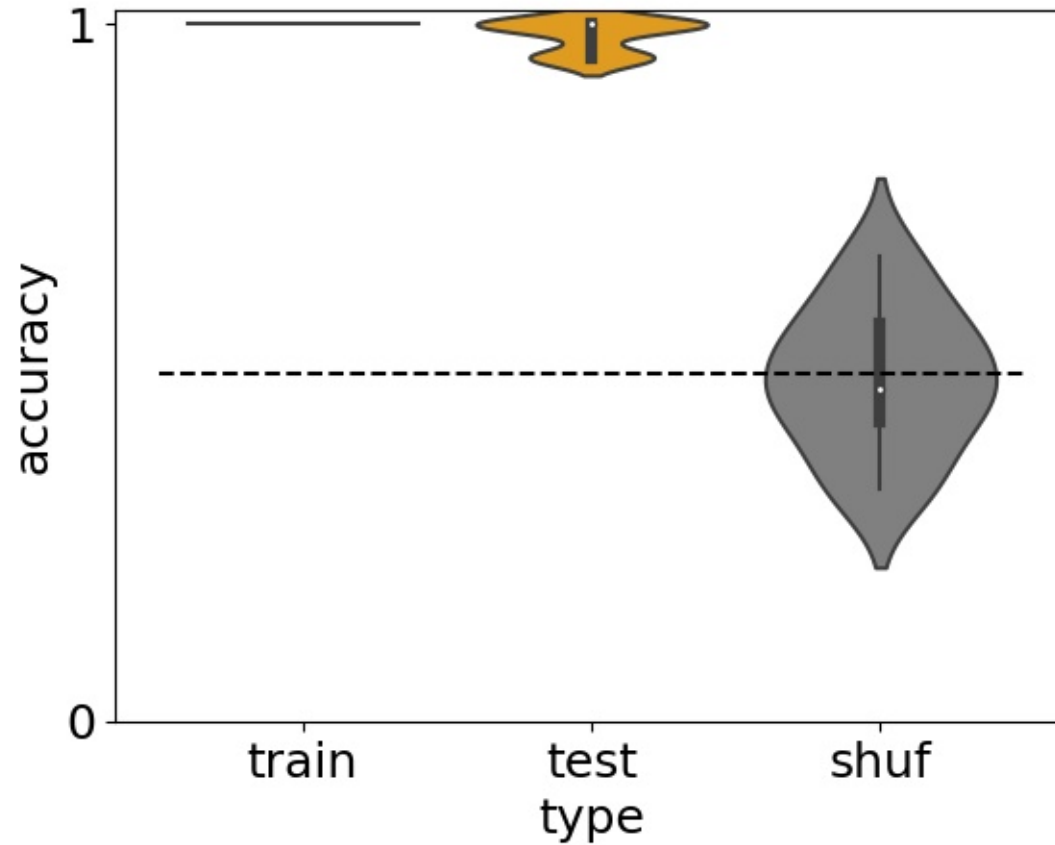
Reliability
across
folds/splits



Randomizing Labels: Baseline = Chance Level



Train, Test and Baseline Accuracy



- distributions, not just means
- control for overfitting

Practice

- Basic exercises in nb_supervised_learning
- Classification of subjects/tasks using data_fMRI
- Try also datasets in scikit-learn (<https://scikit-learn.org/stable/datasets.html>)

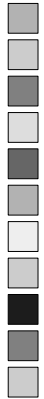
Part 3: Supervised Learning

- Basics
 - Cross-validation
 - Classifiers: basics
 - Hyperparameter tuning with nested cross-validation
 - Feature selection (RFE)
- Application to real datasets
- **Training of logistic regression; autodifferentiation**

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

input

x_i



w_i

output

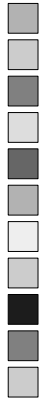
$$y = \phi \left(\sum_i w_i x_i \right)$$

sigmoid
function

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

input

x_i



w_i

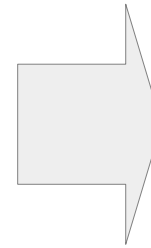
output

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

sigmoid
function

target

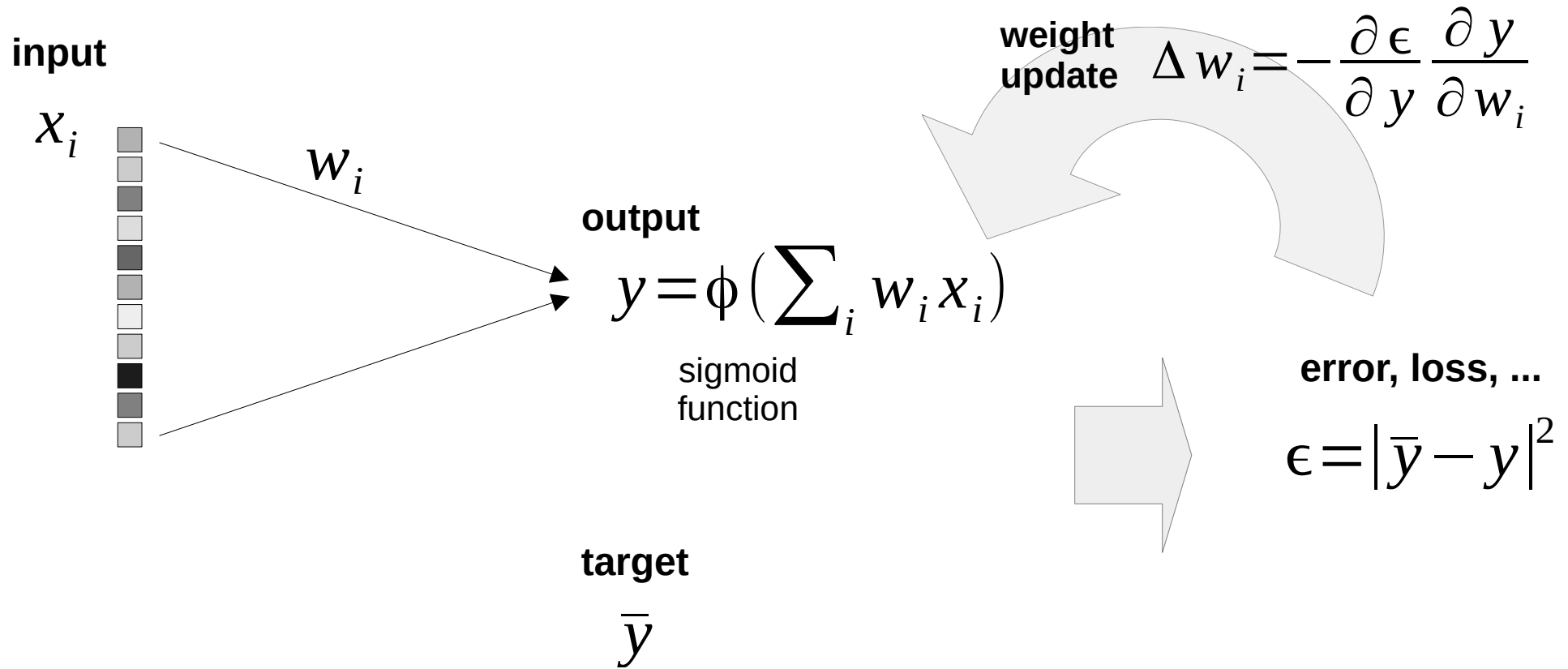
\bar{y}



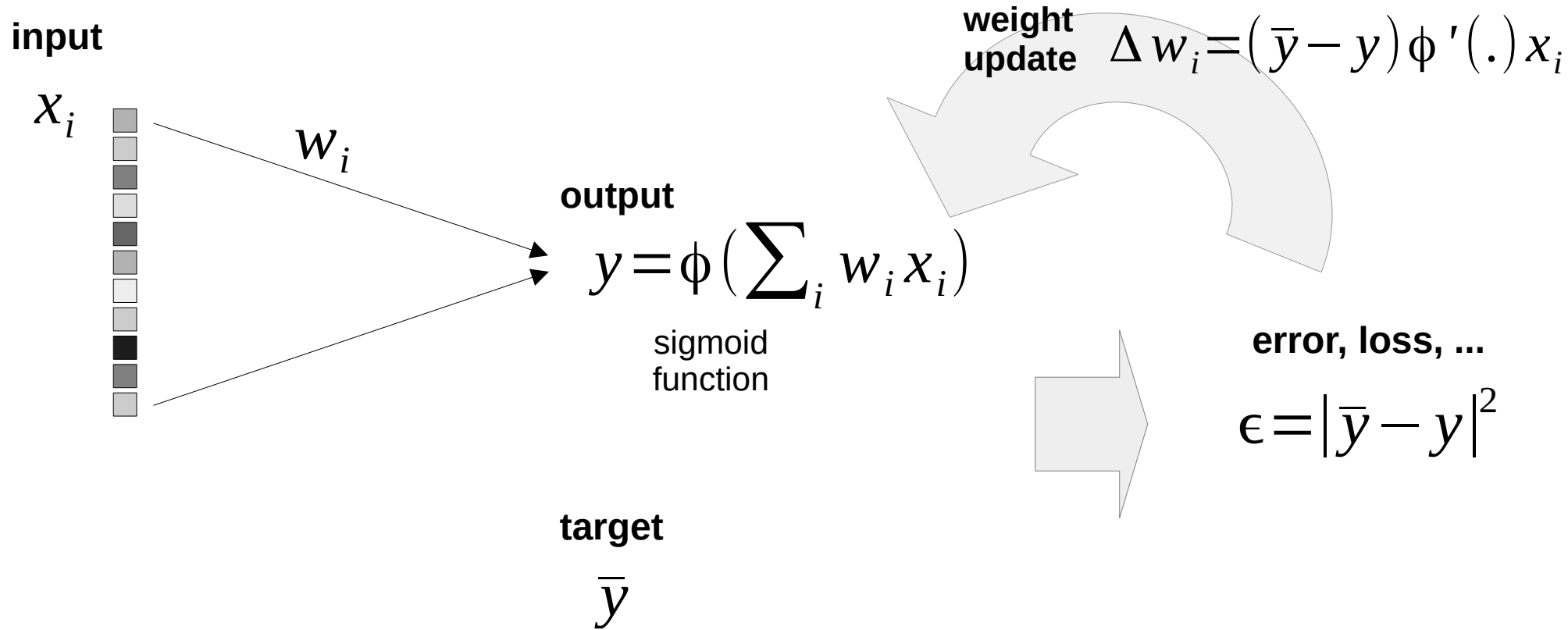
error, loss, ...

$$\epsilon = |\bar{y} - y|^2$$

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



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



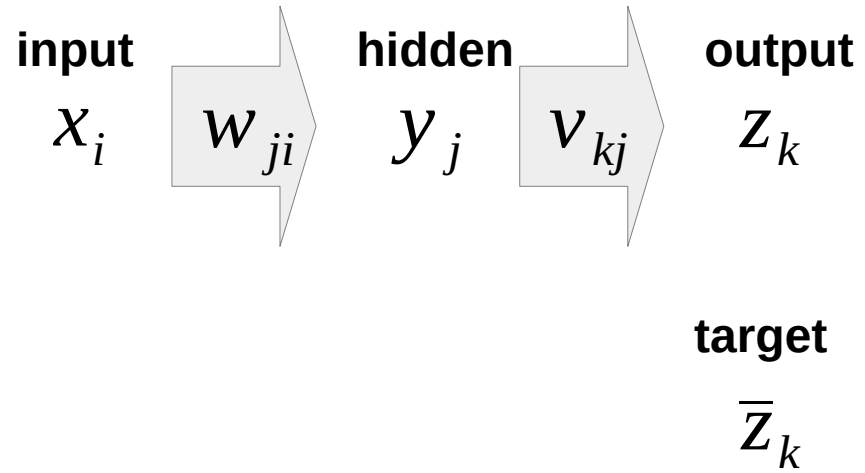
Logistic Regression (a.k.a. Nonlinear Perceptron)

- gradient descent
- depends on loss function and activation function ϕ

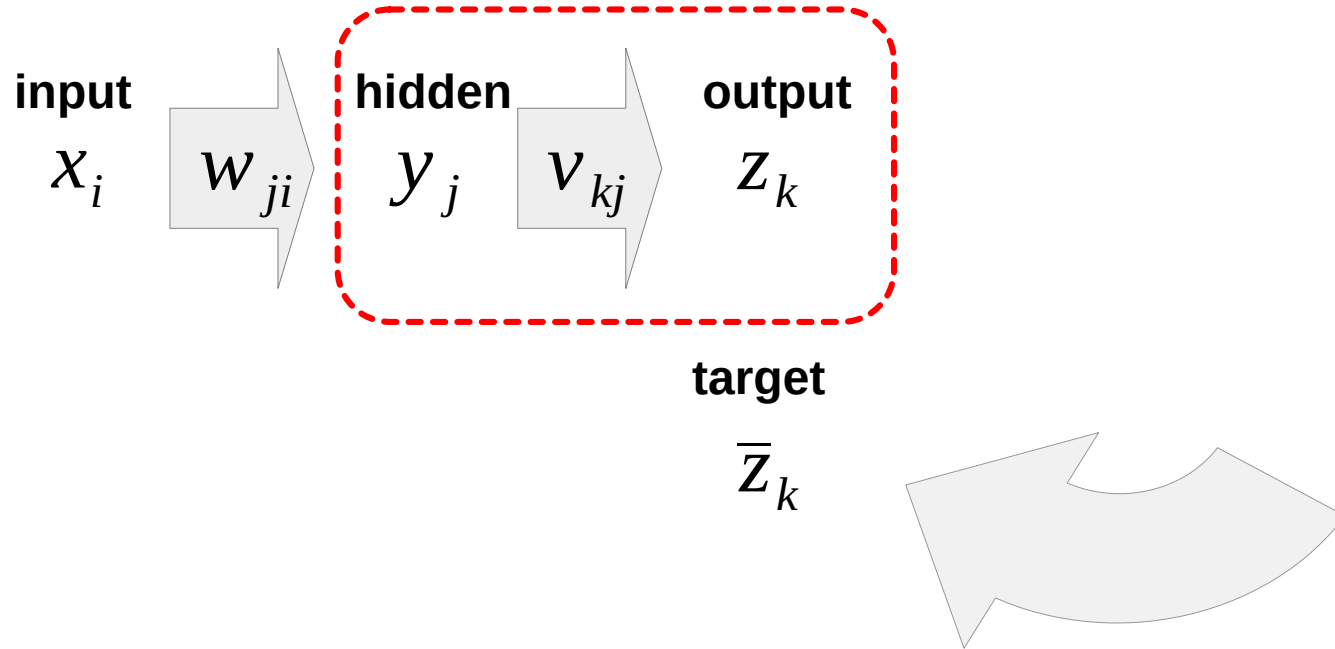
Logistic Regression (a.k.a. Nonlinear Perceptron)

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

Two-Layer Perceptron



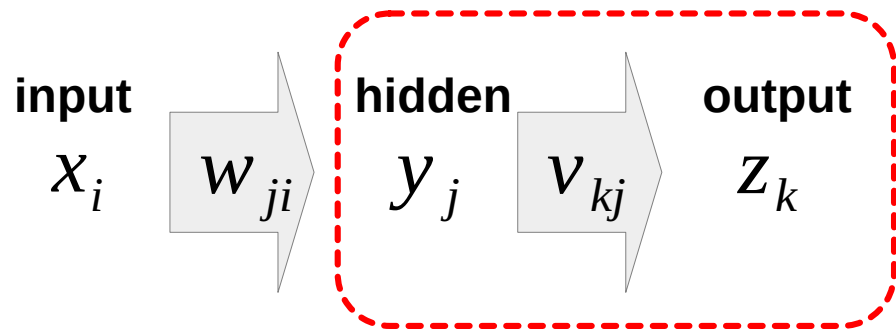
Two-Layer Perceptron



loss

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

Two-Layer Perceptron



target

$$\bar{z}_k$$

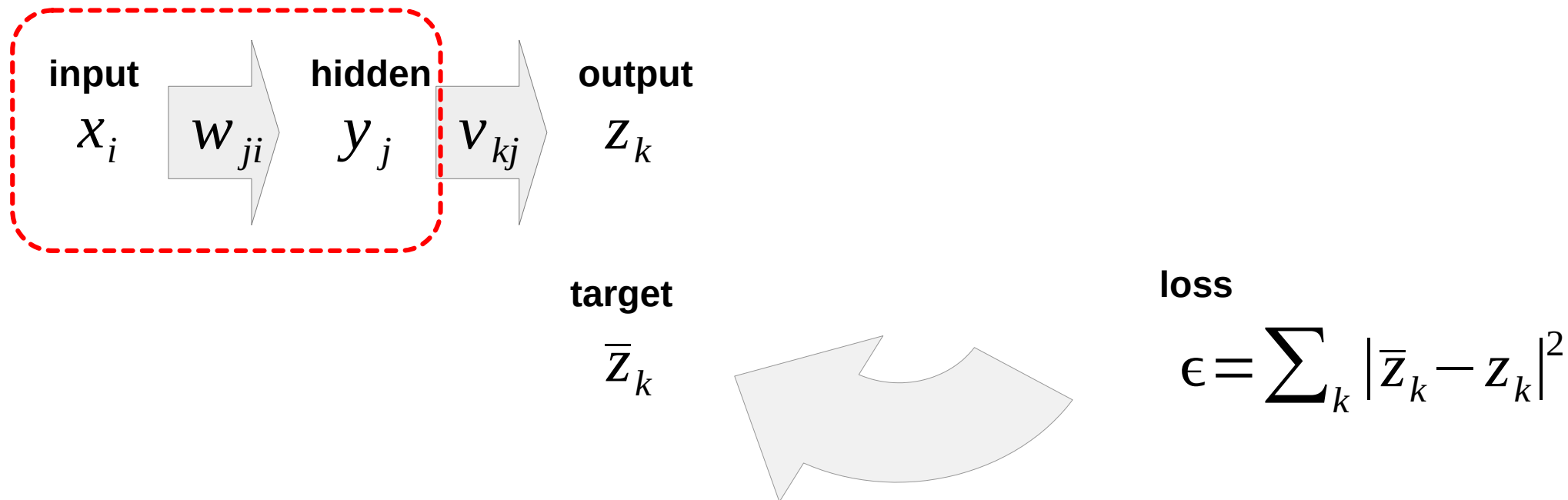
loss

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

weight
update

$$\Delta v_{kj} = (\bar{z}_k - z_k) \phi'(\phi^{-1}(y_j)) y_j$$

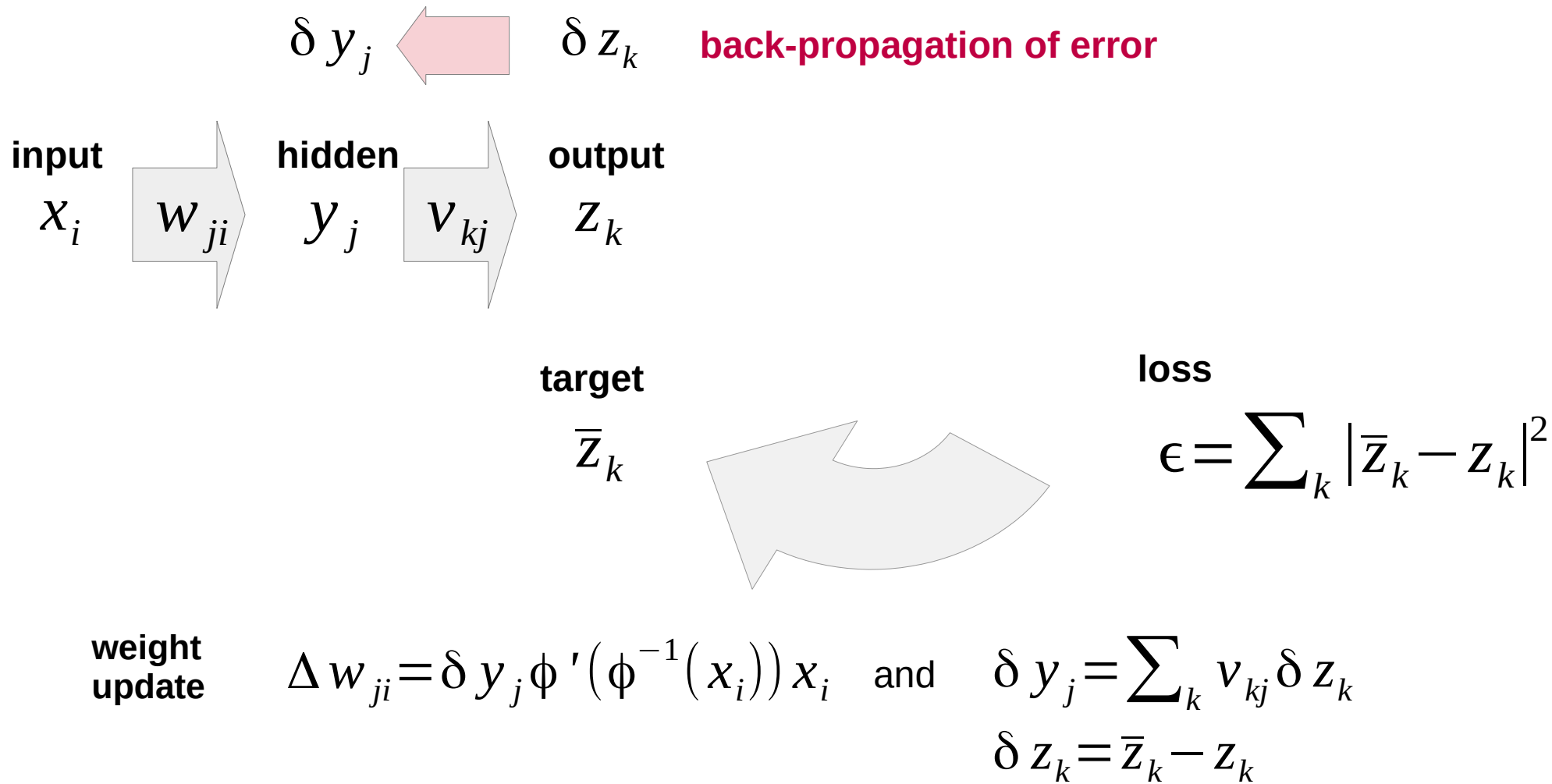
Two-Layer Perceptron



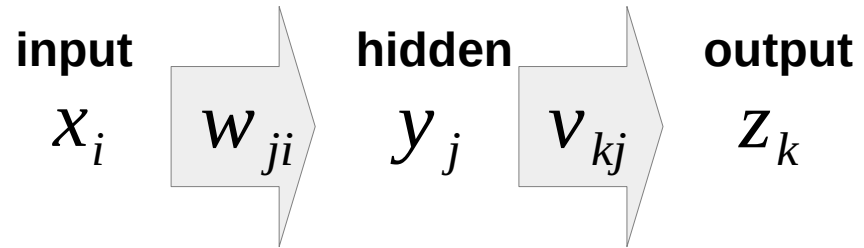
**weight
update**

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

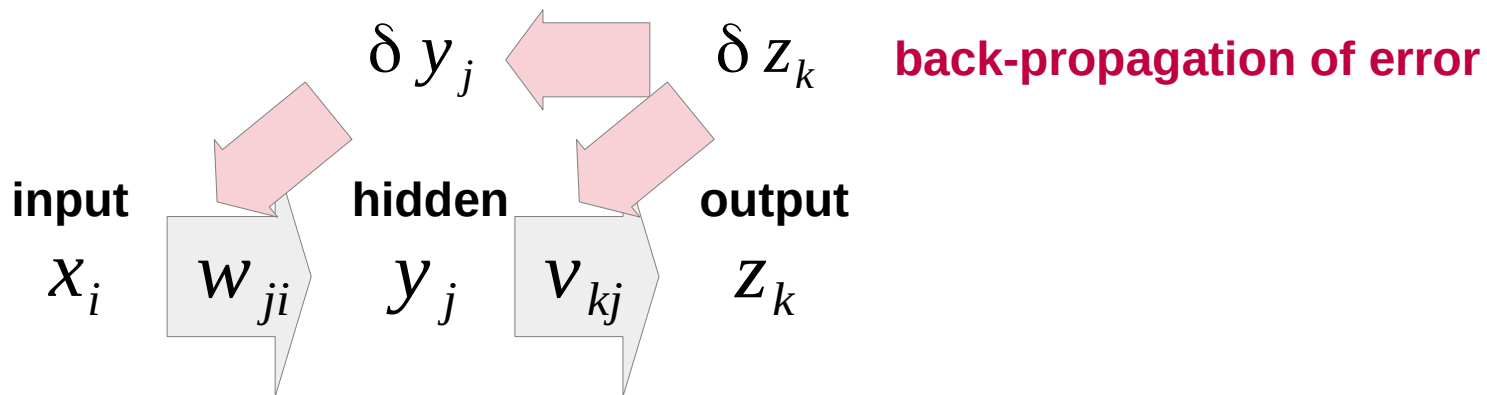
Two-Layer Perceptron



Autodifferentiation



Autodifferentiation



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



Practice

- Tutorial JAX
- Autoencoder, convolutional network in JAX