

# Supervised Learning

- Basics
  - Cross-validation
  - Classifiers: example of logistic regression
  - Hyperparameter tuning with nested cross-validation
  - Feature selection (RFE)
- Application to synthetic and real datasets: library scikit-learn

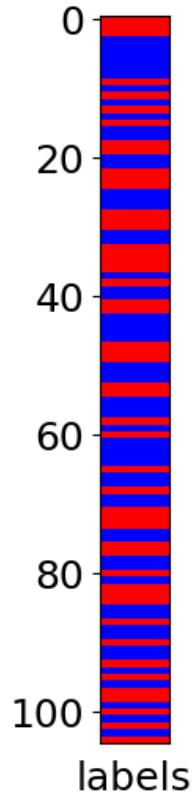
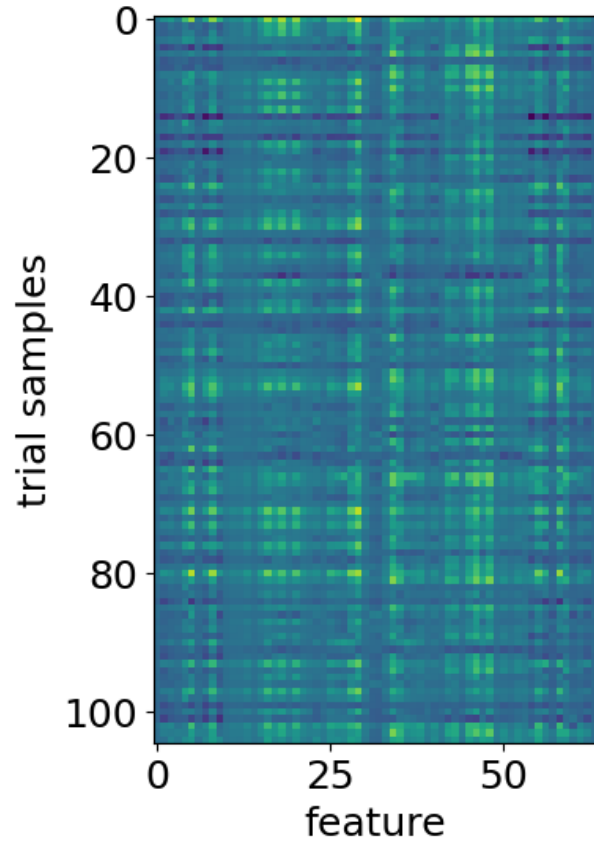
# References

- Articles and reviews:
  - Bishop (2006) *Machine learning and Pattern Recognition*; [https://www.cs.uoi.gr/~arly/courses/ml/tmp/Bishop\\_book.pdf](https://www.cs.uoi.gr/~arly/courses/ml/tmp/Bishop_book.pdf)
  - scikit-learn documentation: <https://scikit-learn.org/>
- To go further on model-based classification:
  - Gilson et al. (2020) *Neuroimage*: model-based analysis of whole-brain dynamics based on fMRI data

# Supervised Learning

- Basics
  - **Cross-validation**
  - Classifiers: example of logistic regression
  - Hyperparameter tuning with nested cross-validation
  - Feature selection (RFE)
- Application to synthetic and real datasets: library scikit-learn

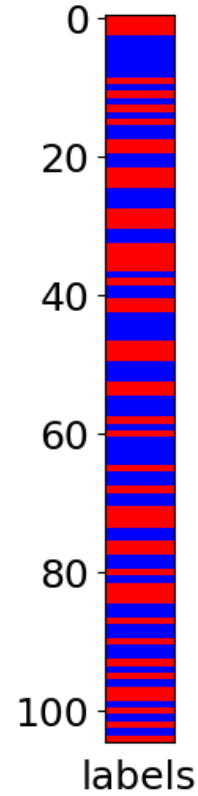
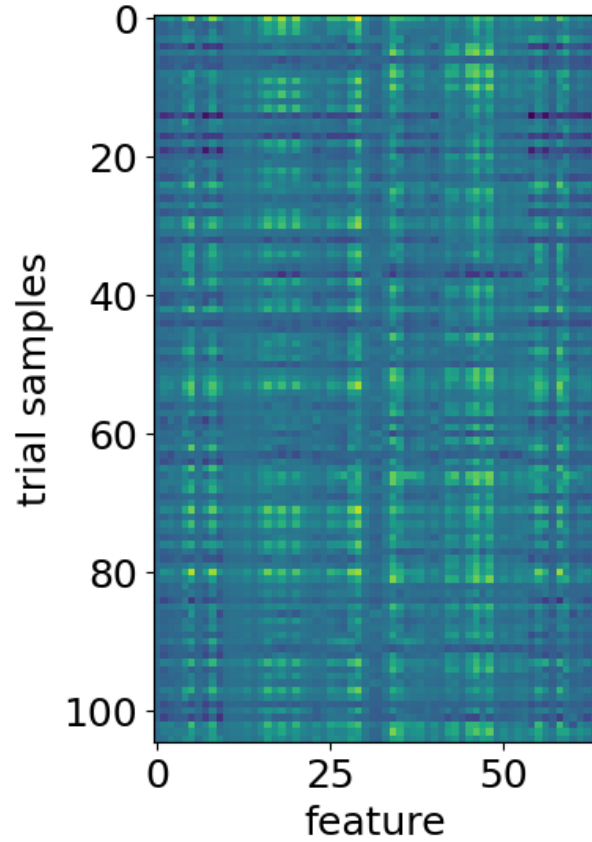
# Cross-Validation



whole  
dataset



# Cross-Validation

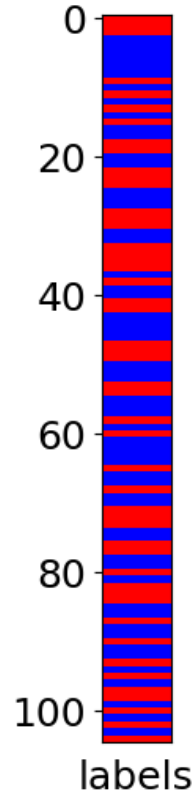
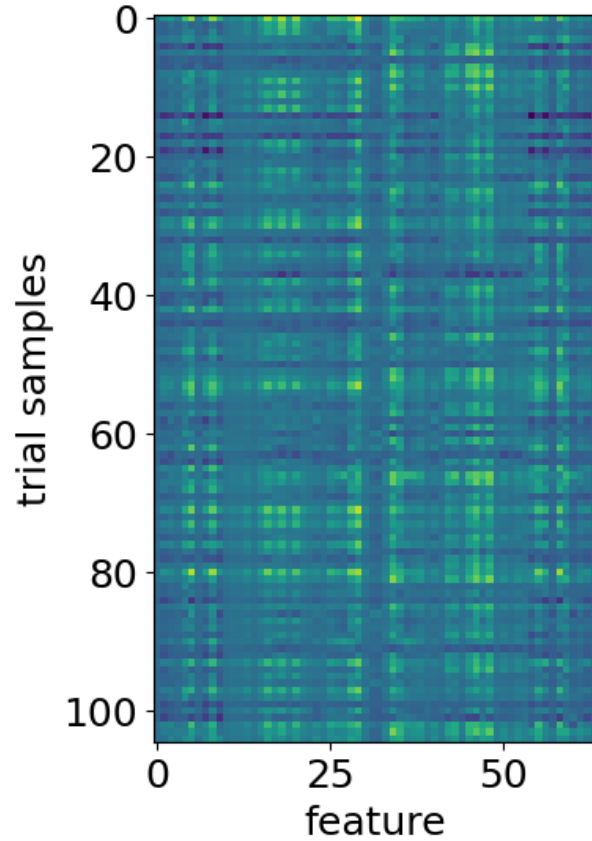


test set (20%)

train set (80%)



# Cross-Validation



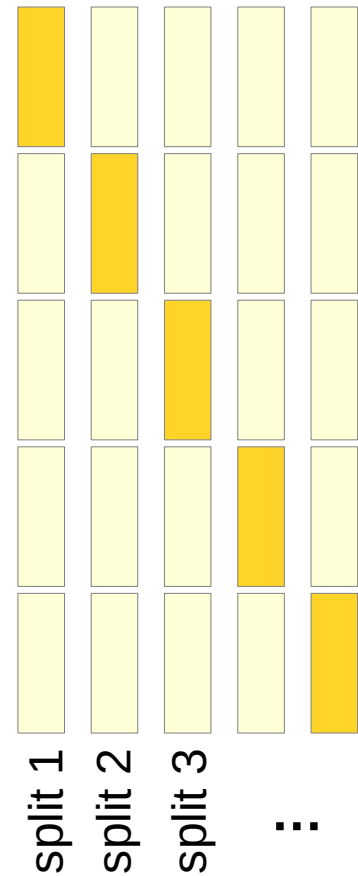
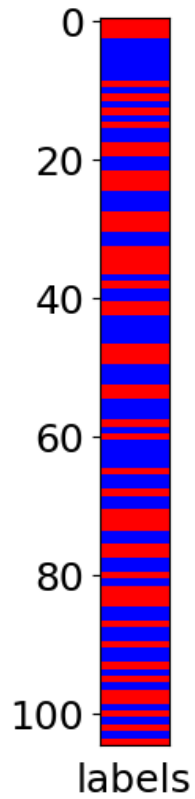
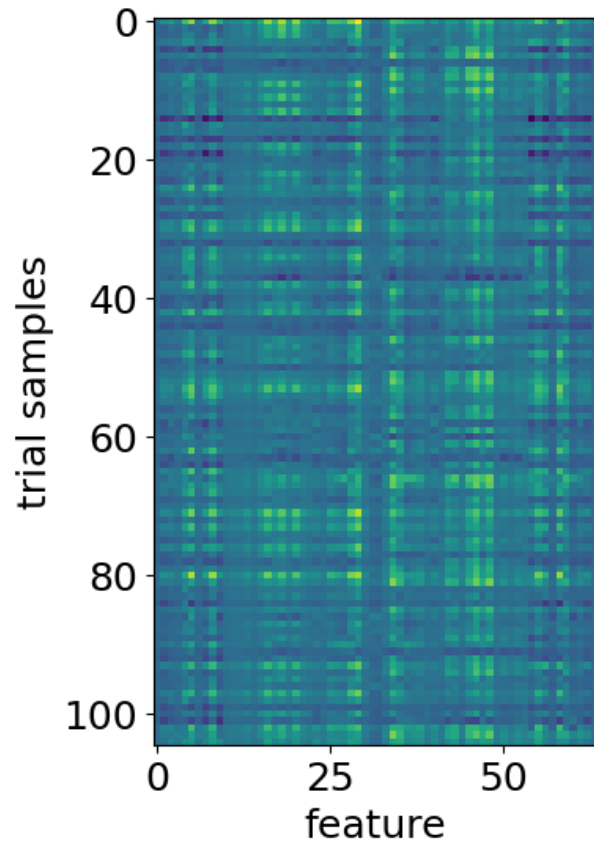
test set (20%)

train set (80%)

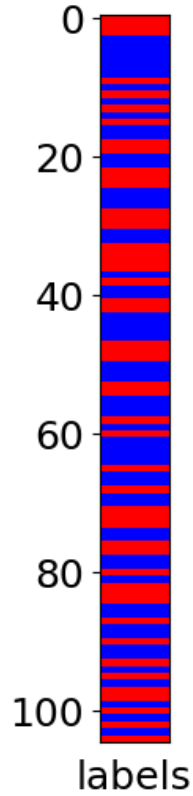
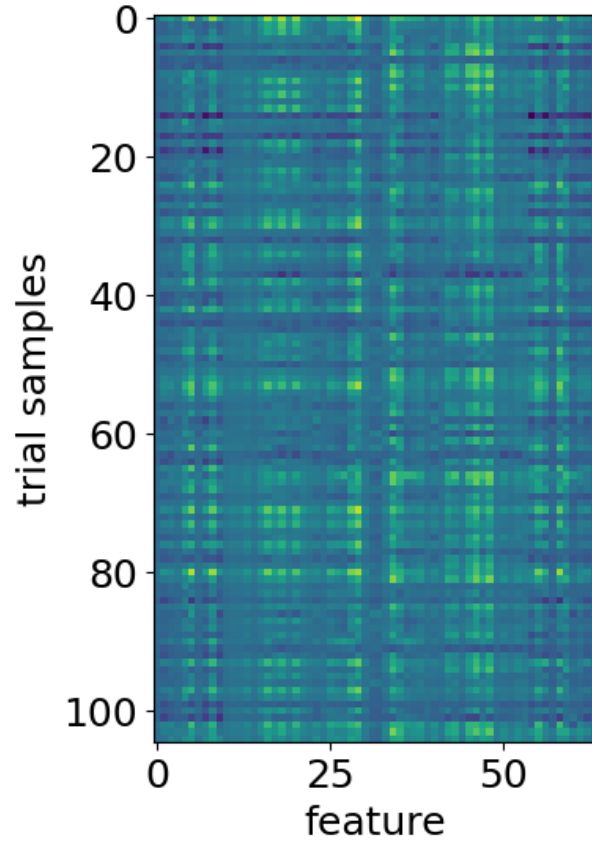


**test accuracy:**  
generalizability  
to “new”  
(unseen) data

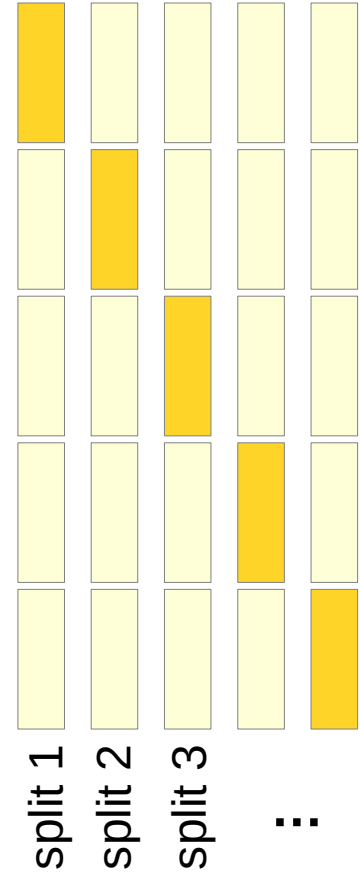
# Cross-Validation



# Cross-Validation

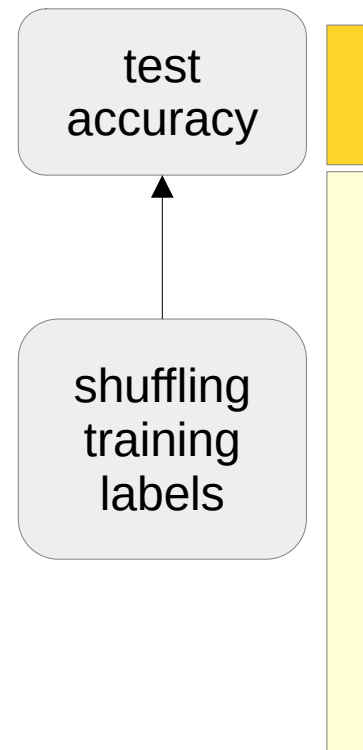
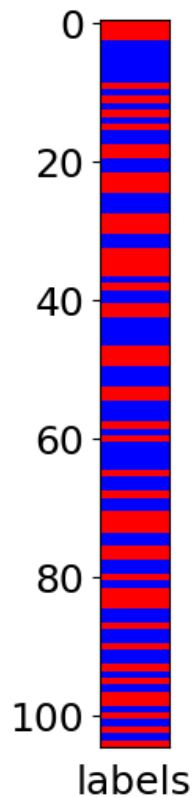
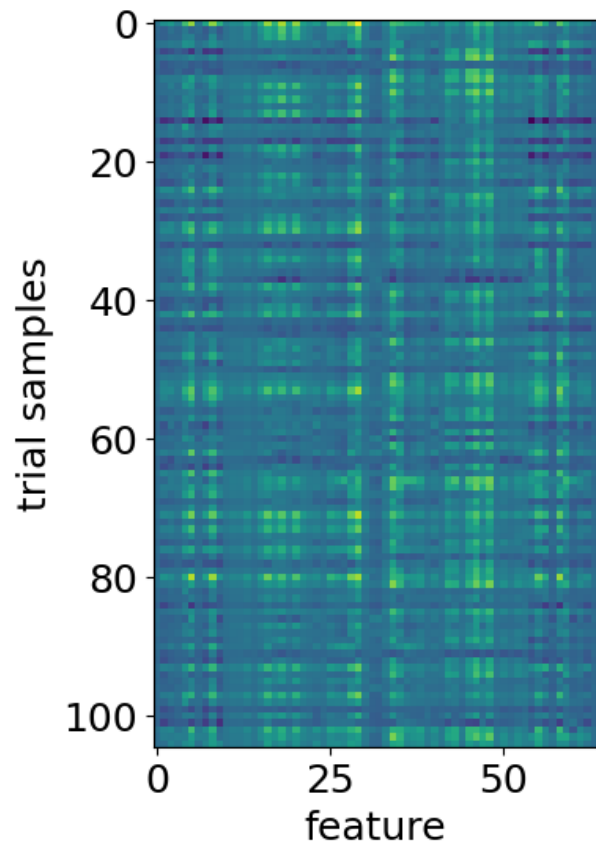


Reliability  
across  
folds/splits

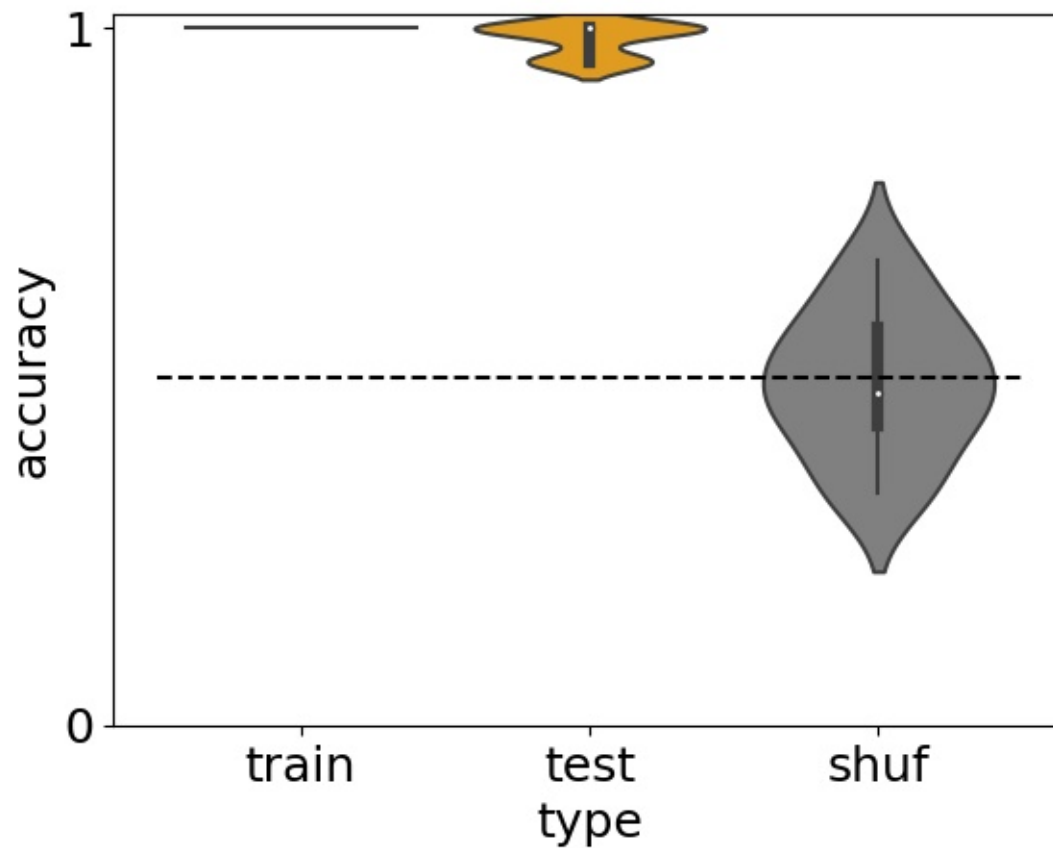




# Randomizing Labels: Baseline = Chance Level



# Train, Test and Baseline Accuracy



- distributions of accuracies, not just mean accuracy
- control for overfitting
- “effect size” of classification

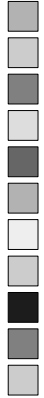
# Class 3: Supervised Learning

- Basics
  - Cross-validation
  - **Classifiers: example of logistic regression**
  - Hyperparameter tuning with nested cross-validation
  - Feature selection (RFE)
- Application to synthetic and real datasets: library scikit-learn

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

input

$x_i$



$w_i$

output

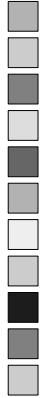
$$y = \phi \left( \sum_i w_i x_i \right)$$

sigmoid  
function

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

input

$x_i$



$w_i$

output

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

sigmoid  
function

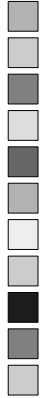


red class

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

input

$x_i$



$w_i$

output

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

sigmoid  
function

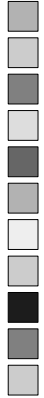


blue class

# 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

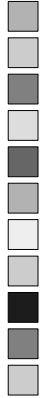
target

$\bar{y}$

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

input

$x_i$



$w_i$

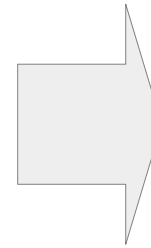
output

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

sigmoid  
function

target

$\bar{y}$

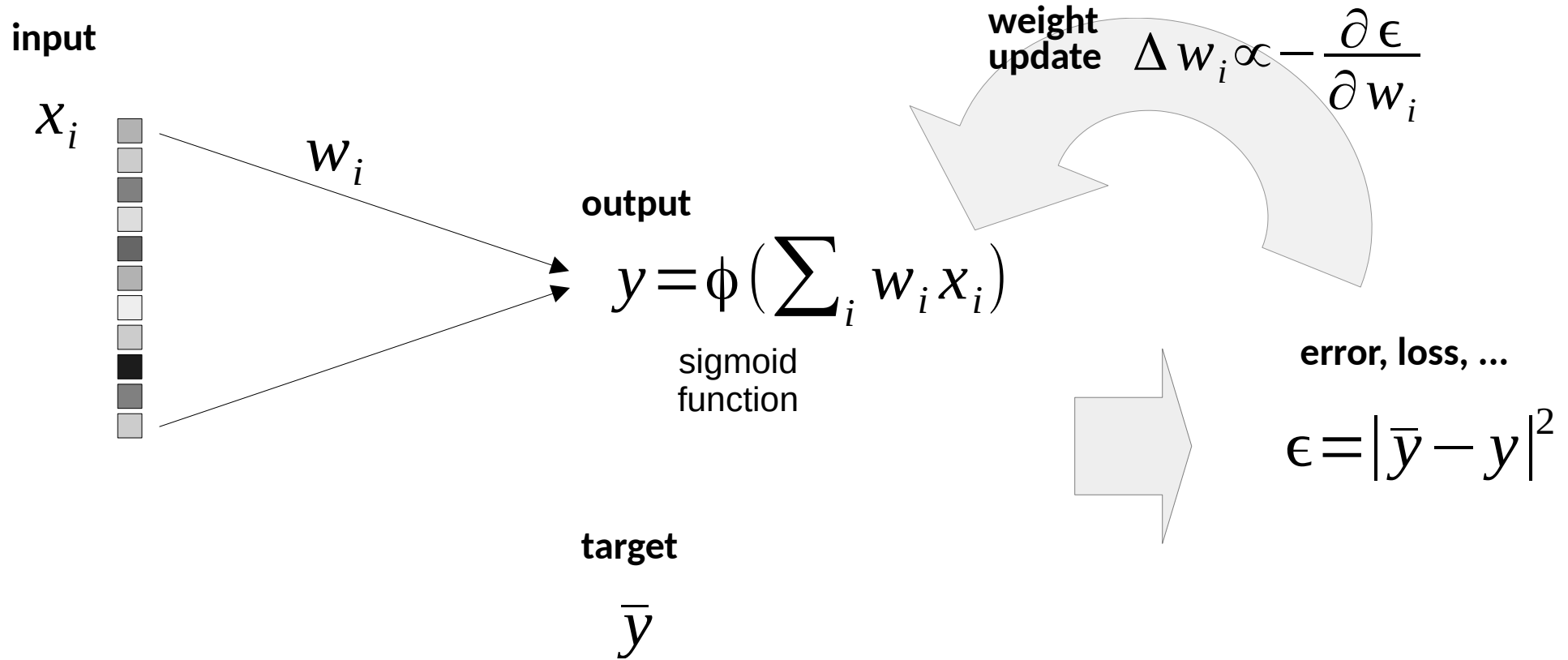


error, loss, ...

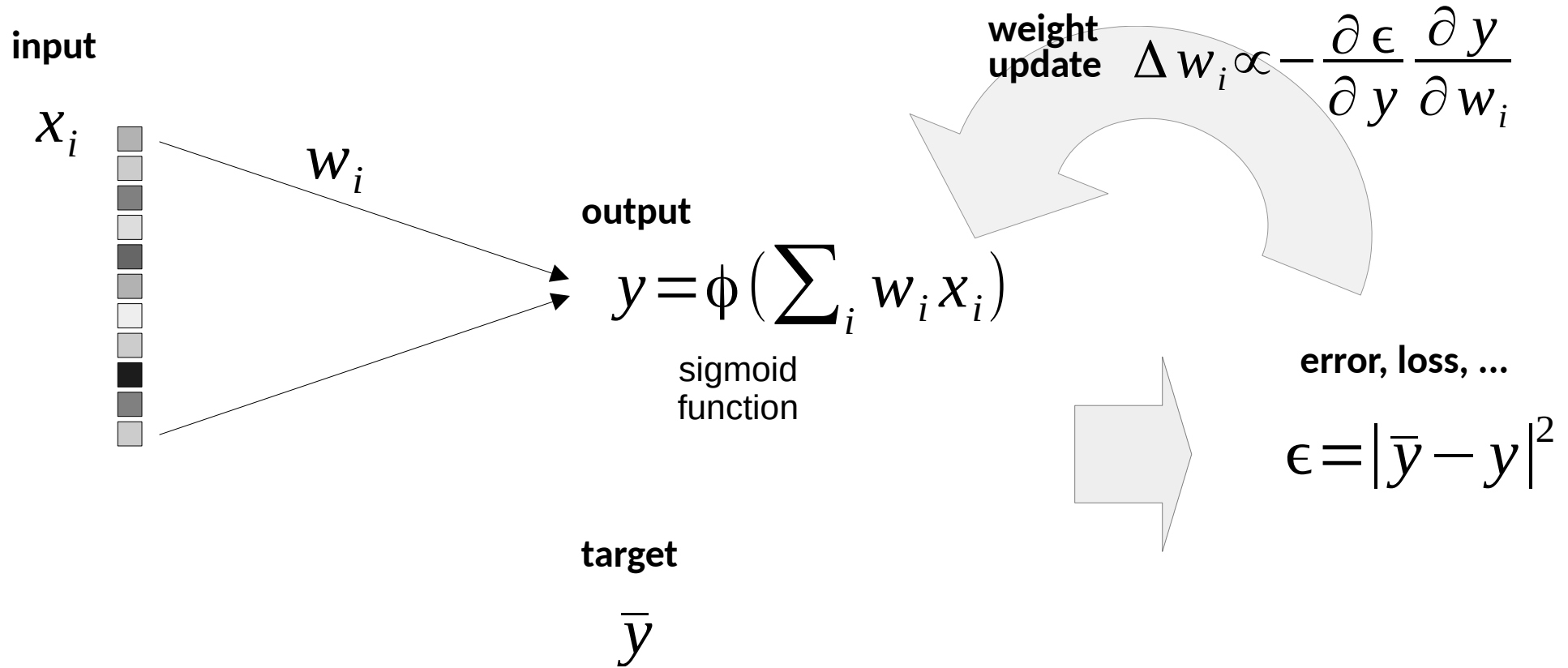
$$\epsilon = |\bar{y} - y|^2$$



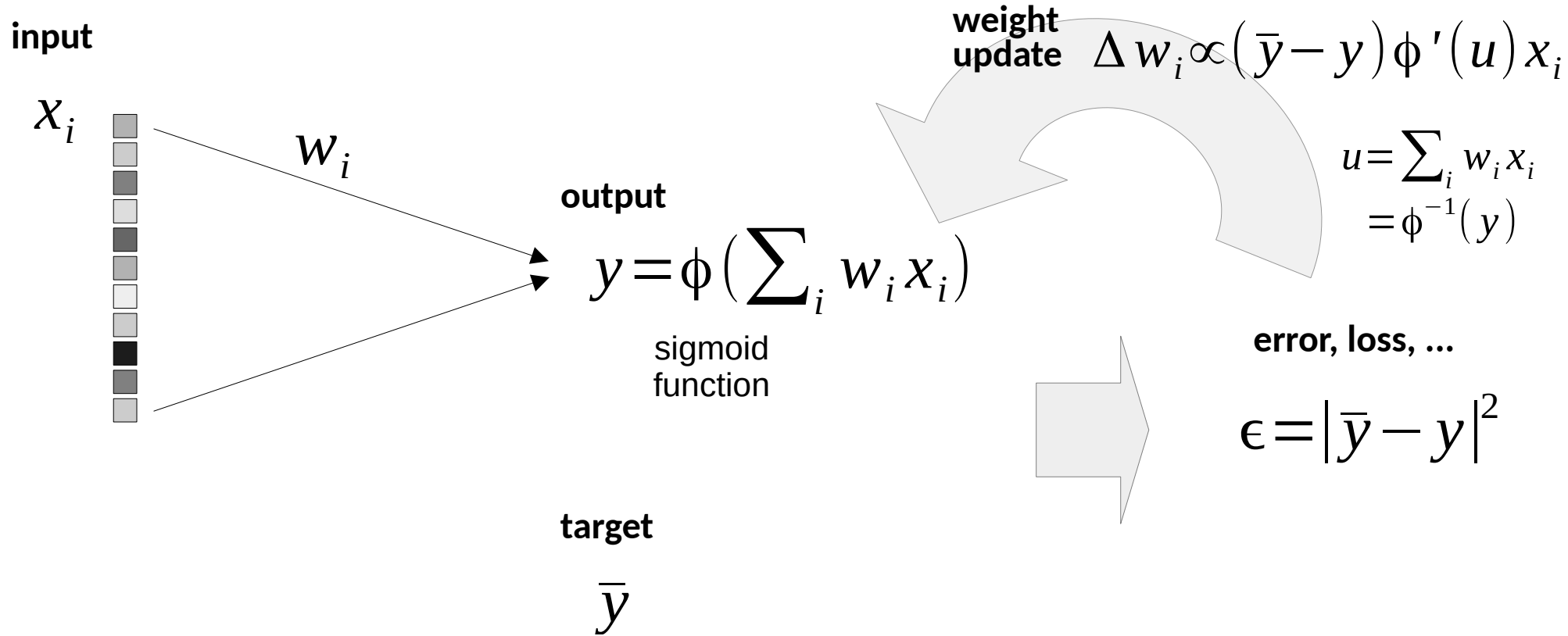
# Logistic Regression (a.k.a. Nonlinear Perceptron)



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



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



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

- gradient descent to optimize weights
- depends on loss function and activation function  $\phi$
- weight update for each input (online) or in batch
- inertia on weight update (e.g. Adam optimizer)

$$\Delta w^{eff} \leftarrow \beta \Delta w^{eff} + (1 - \beta) \Delta w$$

# Class 3: Supervised Learning

- Basics
  - Cross-validation
  - Classifiers: example of logistic regression
  - **Hyperparameter tuning with nested cross-validation**
  - Feature selection (RFE)
- Application to synthetic and real datasets: library scikit-learn

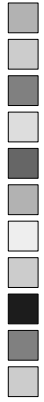
# Enforce Sparsity of Learned Weight Structure

gradient descent:

$$\Delta w_i = (\bar{y} - y) \phi'(\phi^{-1}(y)) x_i$$

input

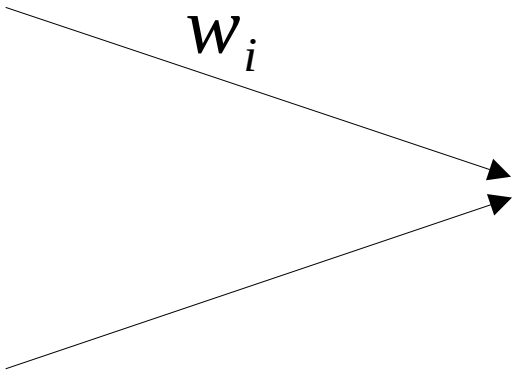
$x_i$



$w_i$

output

$y$



# Enforce Sparsity of Learned Weight Structure

gradient descent:

$$\Delta w_i = (\bar{y} - y) \phi'(\phi^{-1}(y)) x_i - \frac{1}{C} w_i$$

regularization

input

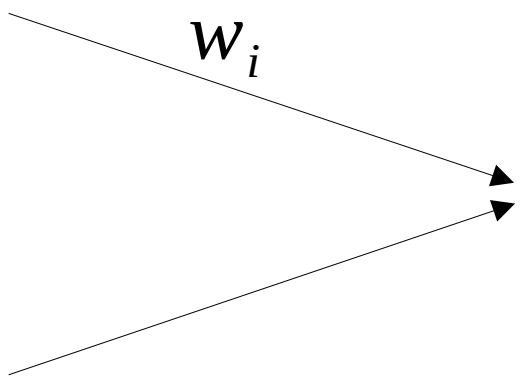
$x_i$



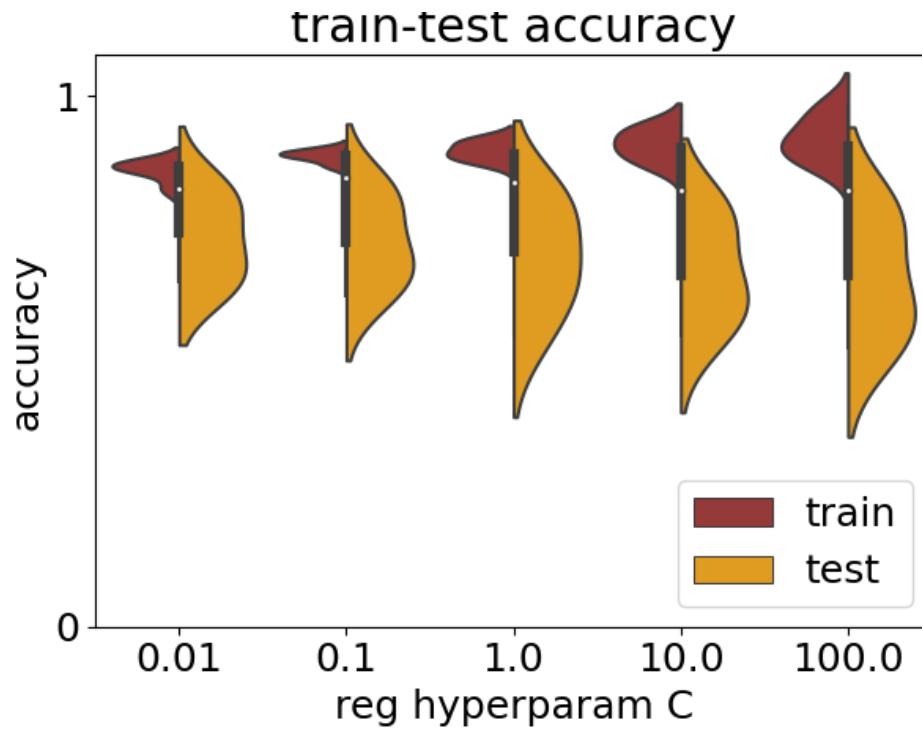
$w_i$

output

$y$



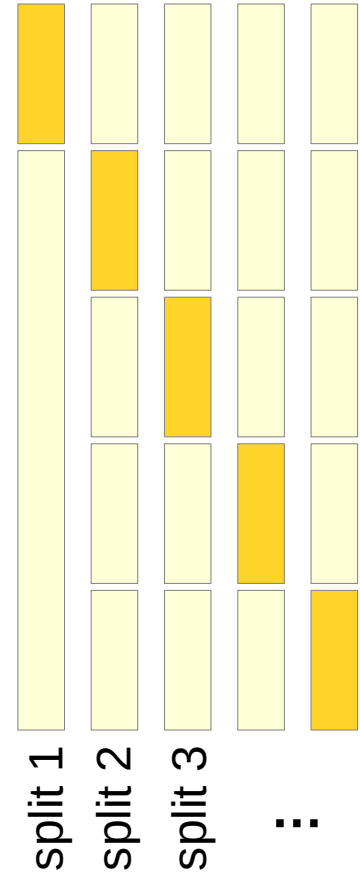
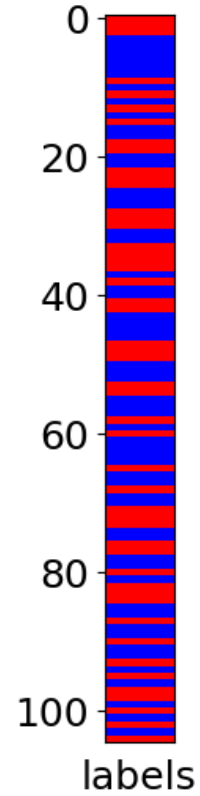
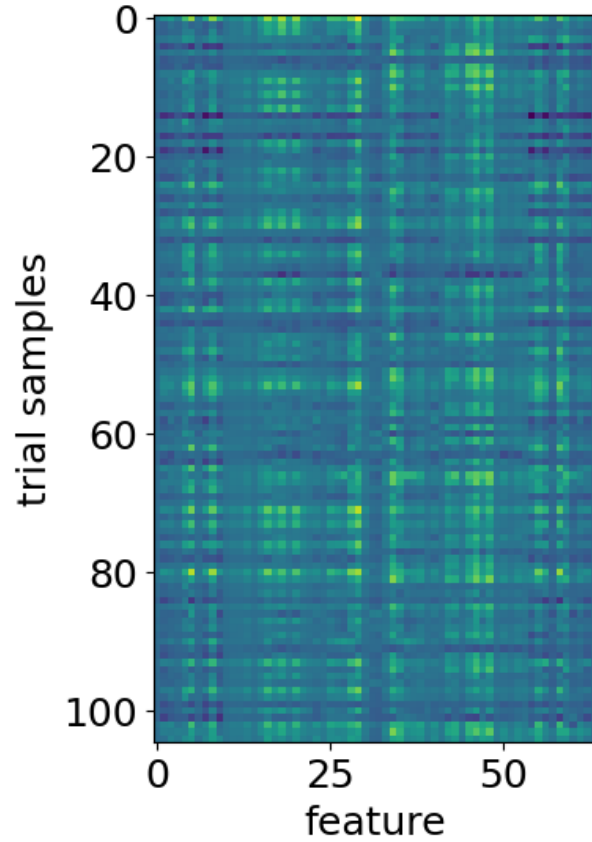
# Importance of Regularization



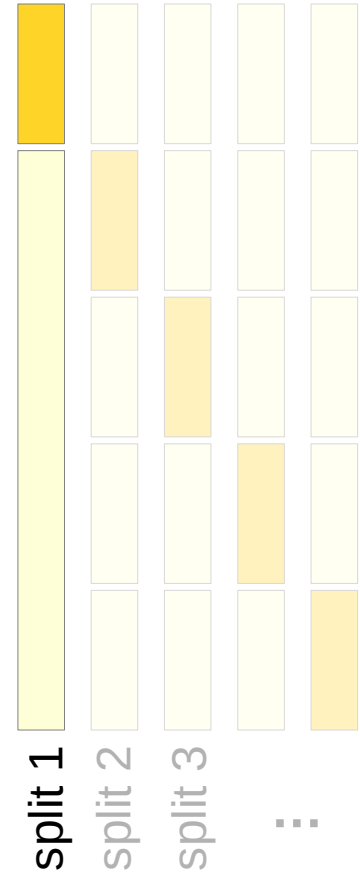
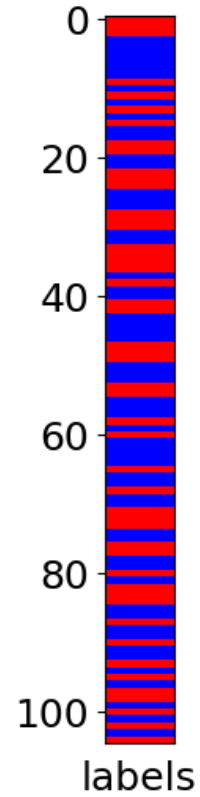
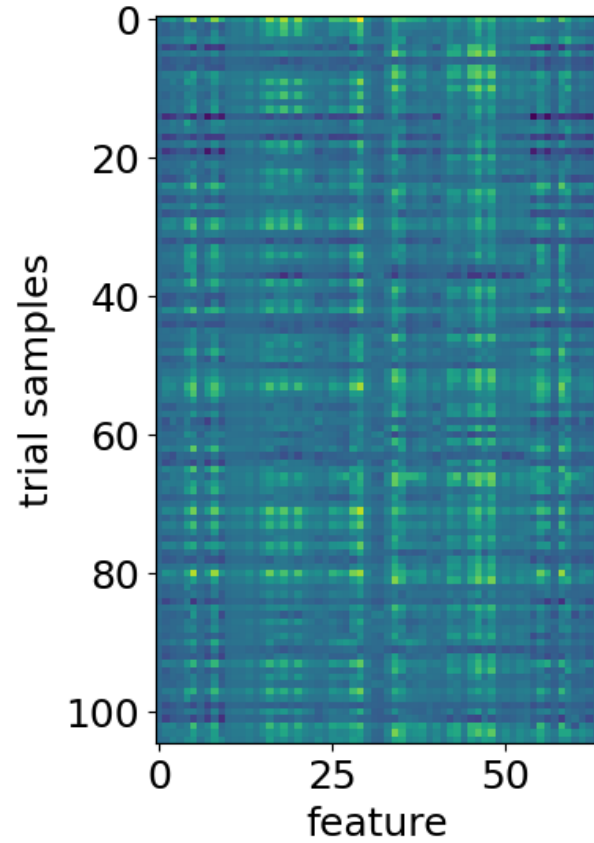
- smaller C implies more sparse
- opposing trends for train and test accuracies
- important to focus on some features in case of large input dimensionality



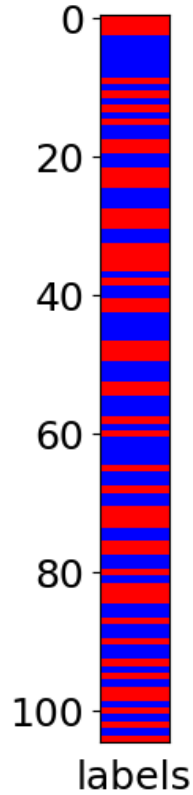
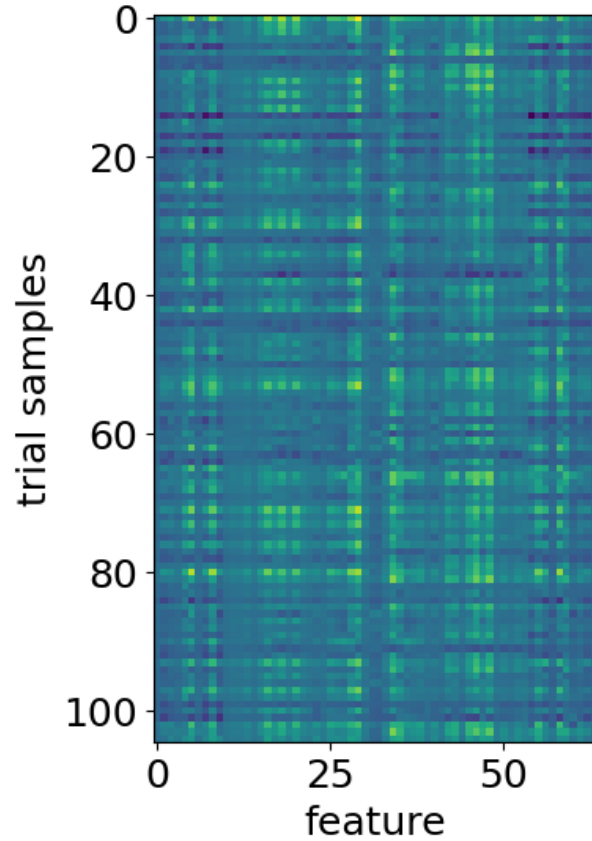
# Nested Cross-Validation for Hyperparameter Tuning



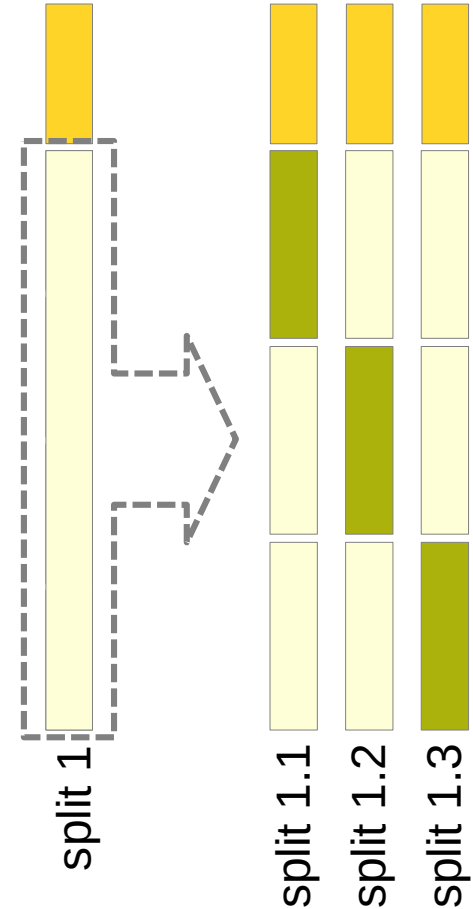
# Nested Cross-Validation for Hyperparameter Tuning



# Nested Cross-Validation for Hyperparameter Tuning



estimate best C  
from training set  
(best performance  
on **validation set**)

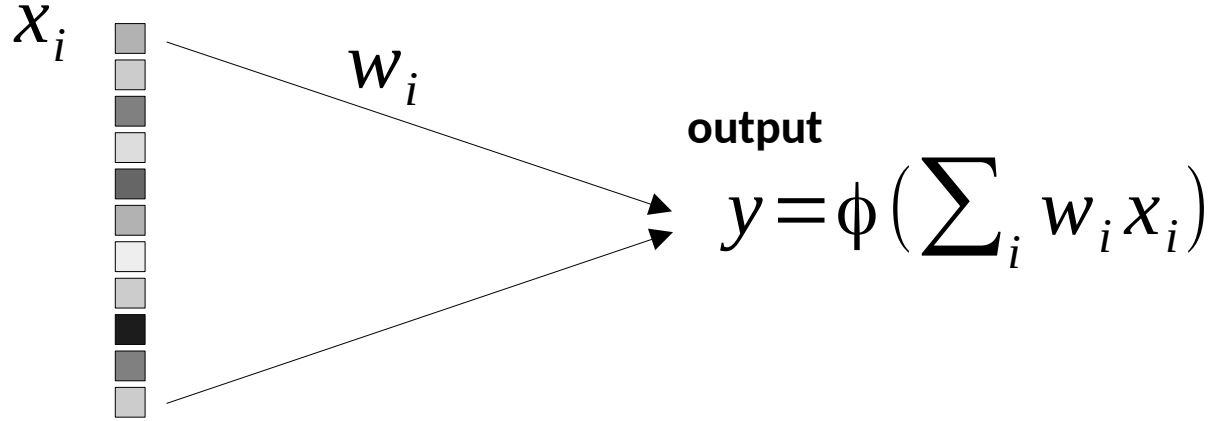


# Class 3: Supervised Learning

- Basics
  - Cross-validation
  - Classifiers: example of logistic regression
  - Hyperparameter tuning with nested cross-validation
  - **Feature selection (RFE)**
- Application to synthetic and real datasets: library scikit-learn

# Recursive Feature Elimination (RFE)

input

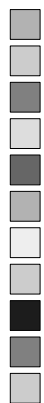


- large weight (in abs value): important feature
- important: needs for rescaling inputs (z-score samples for each feature)
- step-wise removal of weak weights first to identify important features, provides a ranking of feature importance

# Recursive Feature Elimination (RFE)

input

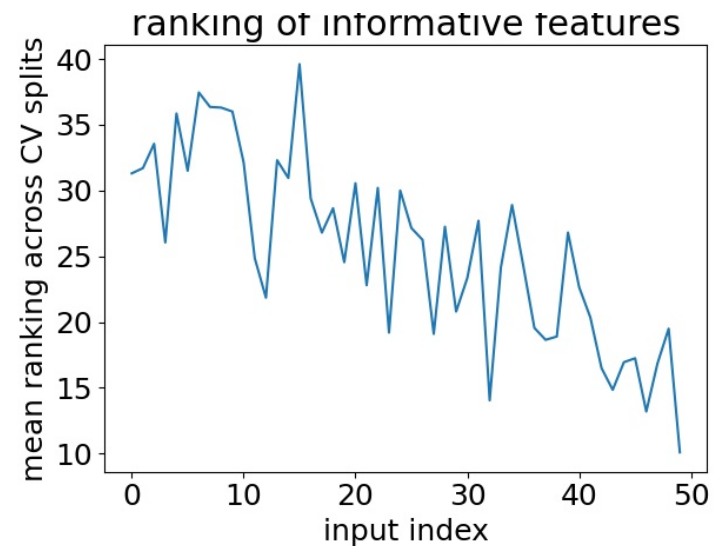
$x_i$



$w_i$

output

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



# Practice

- Basic exercise in *nb\_suplearn\_exercise*
  - also further details on cross-validation (simple and nested),
- Classification of subjects/tasks using data\_fMRI
- Try also datasets in scikit-learn (<https://scikit-learn.org/stable/datasets.html>)