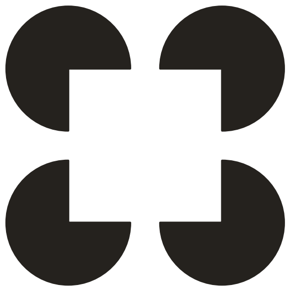
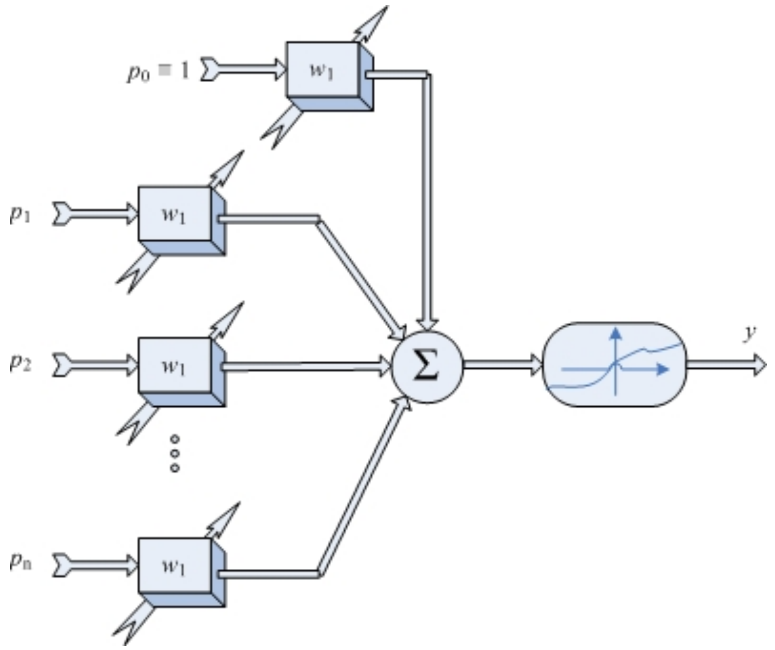
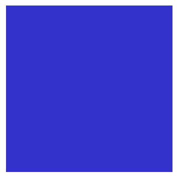
ENE0154 IntelComp – Inteligência Computacional

Additional Topics on Supervised Training of Multi-Layer Perceptron

Prof. Adolfo Bauchspiess

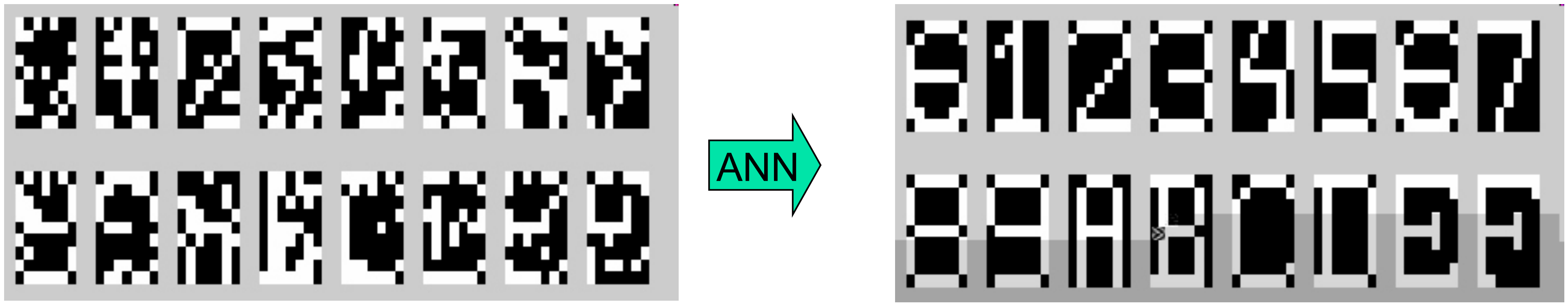




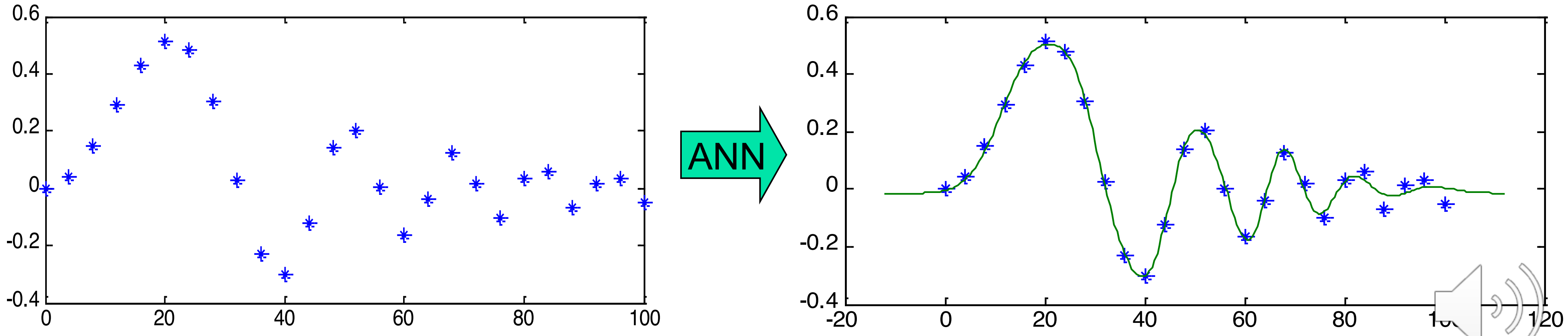


ANN Applications

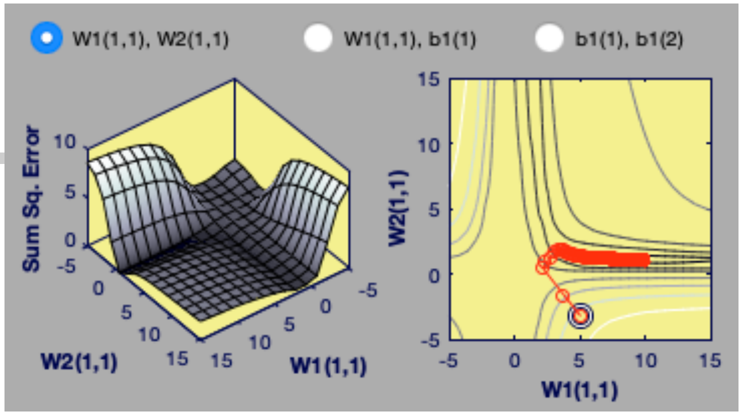
- Pattern Classification



- Function Approximation (non linear)

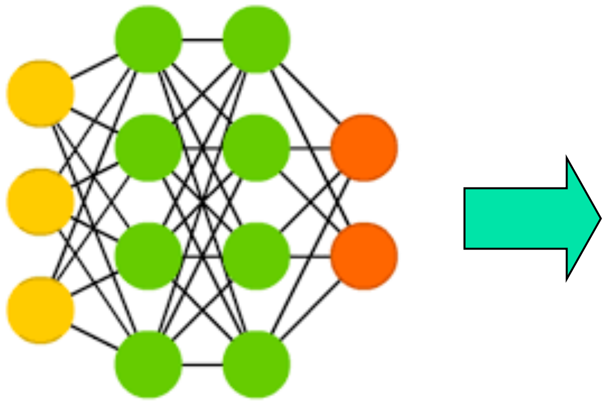


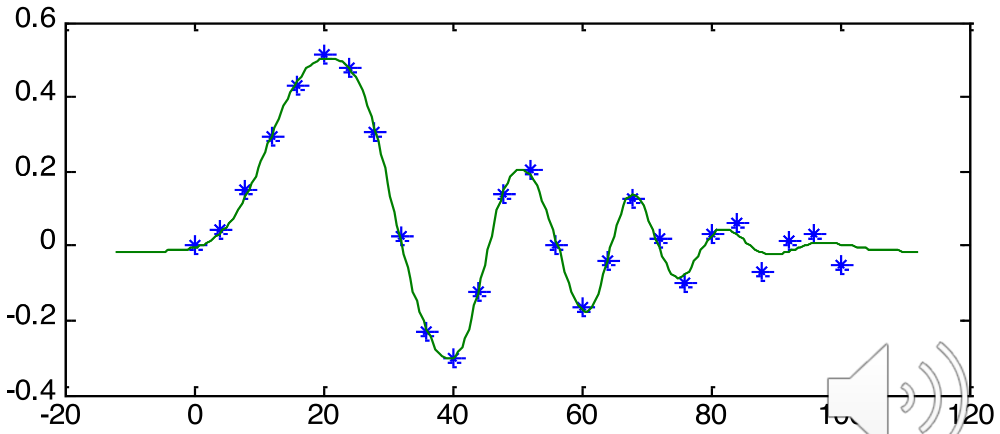
MLP Design



- Data set: train, val, test; #, consistency
- Problem "complexity"

- Topology: # inputs, # outputs, #layers, #neuros/layer
- Activation function: per layer
- Training Algorithm
- Initial Conditions
- Stop Conditions: error goal, #epochs, #gradient

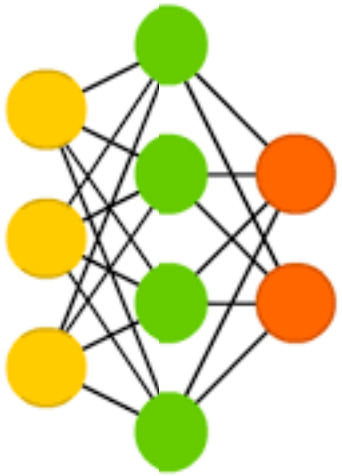
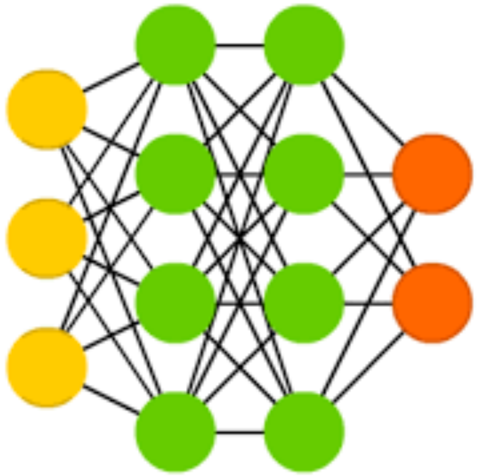
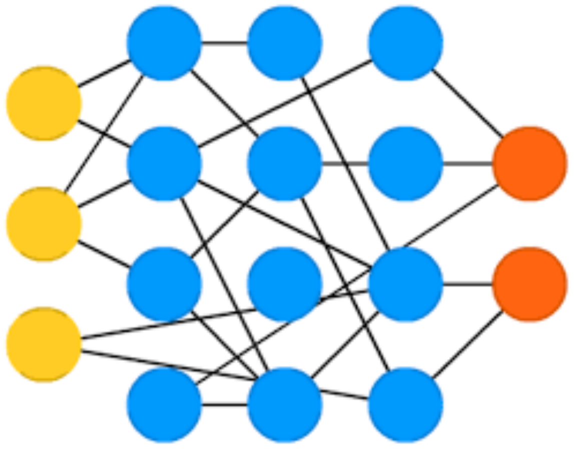




Topology

According to Cybenko's Theorem, 1989,
an ANN with one hidden layer is a universal approximator.

Why then use more layers?
  Sometimes is easier to train.
    Deep Learning: each layer represent different features (inspection).

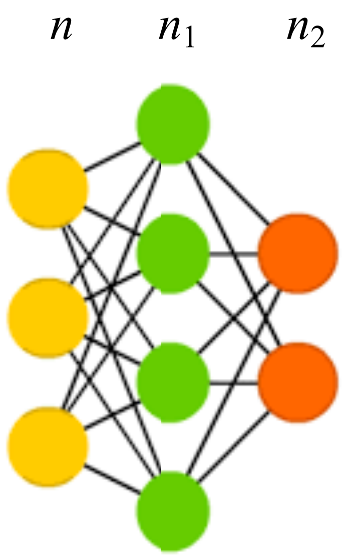




# Heuristics for one Hidden Layer

$$n_1 = 2 \cdot n + 1 \quad \{\text{Kolmogorov method}\}$$

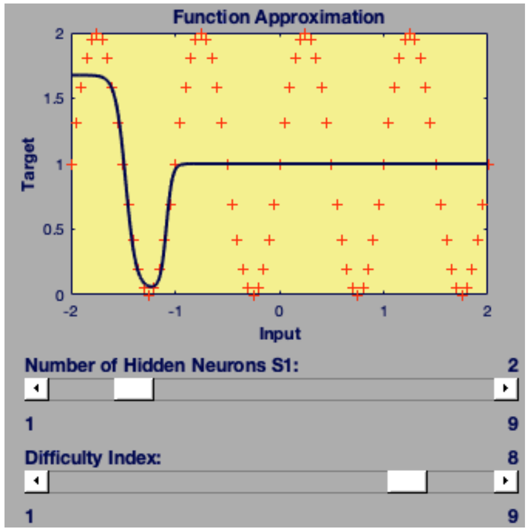
$$2 \cdot \sqrt{n} + n_2 \leq n_1 \leq 2 \cdot n + 1 \quad \{\text{Fletcher-Gloss method}\},$$

$$n_1 = (n + n_2)/2 \quad \text{Classification}$$



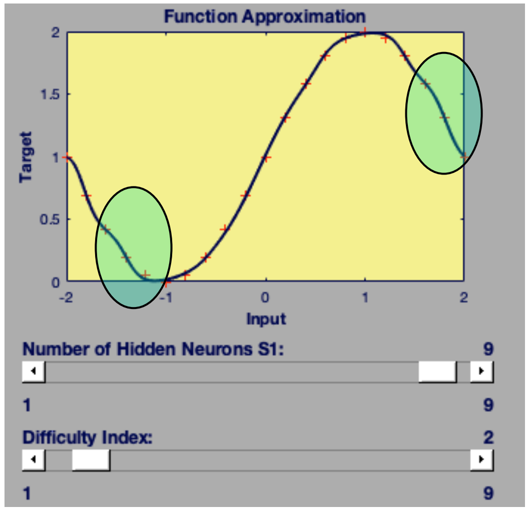
- Underfitting

To few neurons for the problem



- Overfitting

To much neurons for the problem



Train, Test and Validation Sets

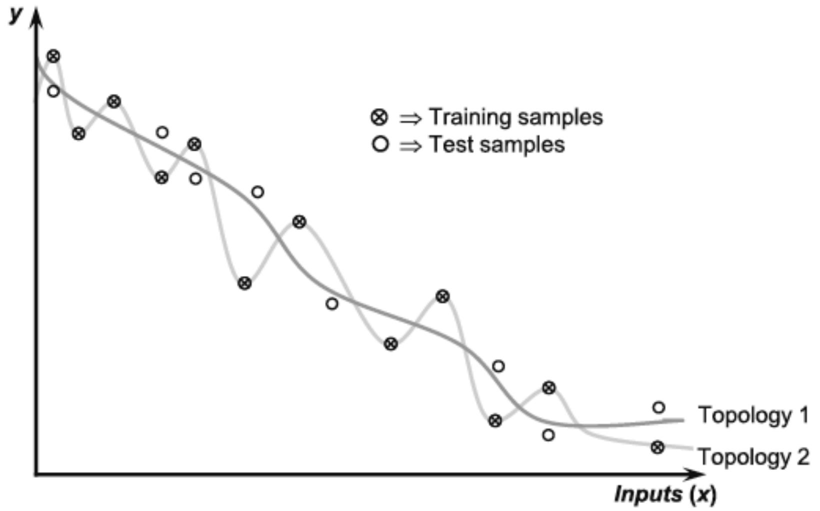
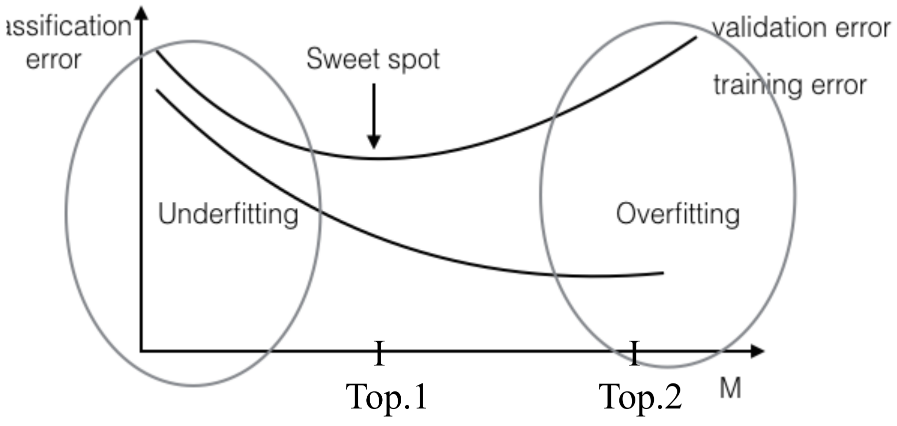
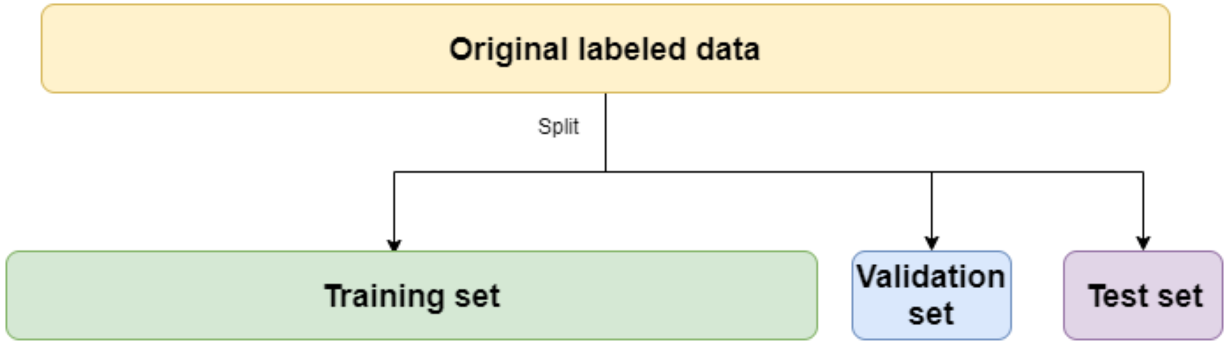


Fig. 5.38 Behavior of MLP networks operating with and without overfitting





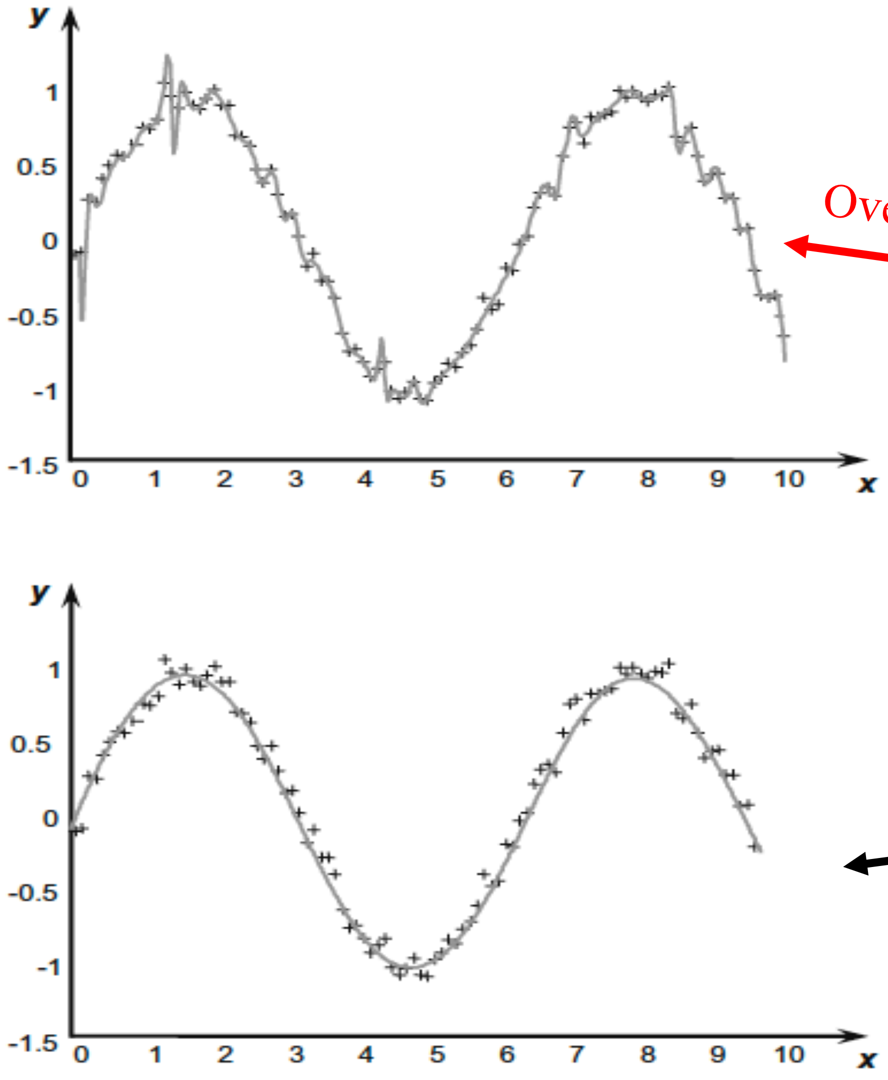
How to split? (Typically 70% 15% 15%)

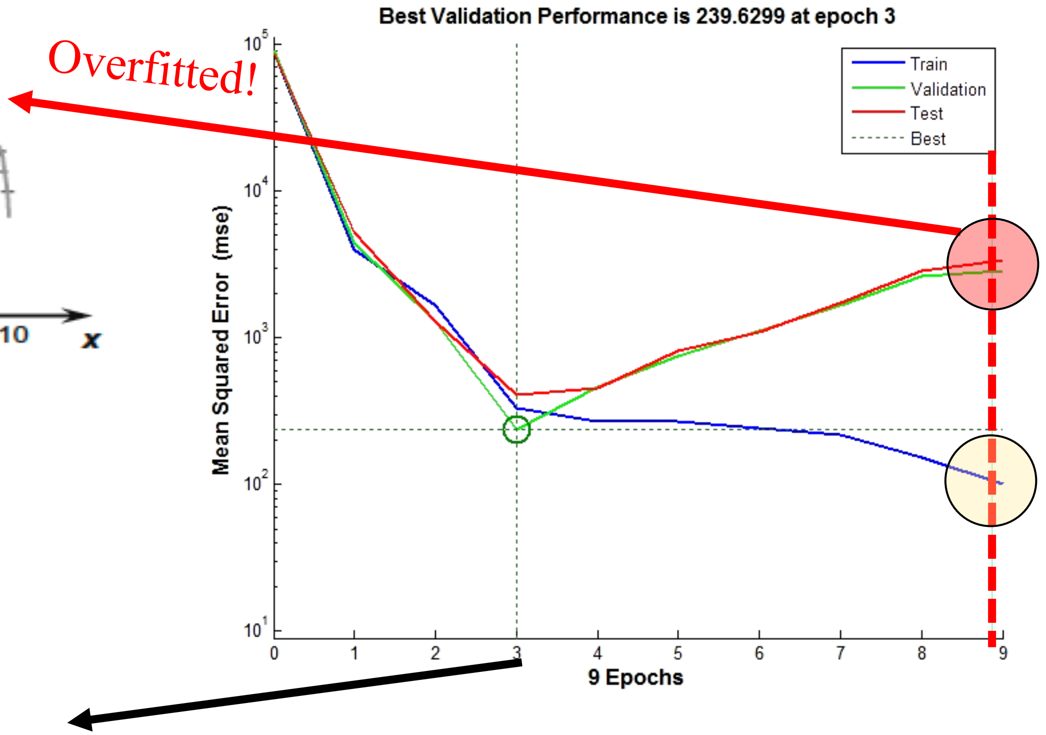
Train: set that changes $W_{ij}^{(k)}$

Val: set used to
prevent overfitting
does not change $W_{ij}^{(k)}$

Test: independent set
not used to train
not used to stop training

Overfitting Example

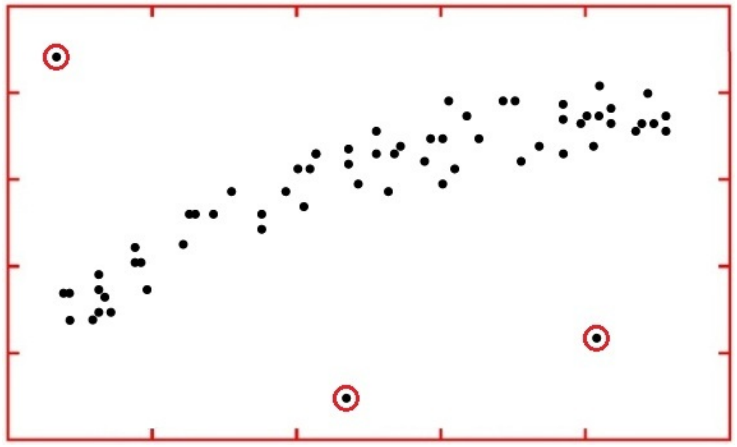




(Illustrative training curves. Not from sine example)

Cross Validation

Reduce data dependency.
E.g., “outliers” – corrupt the training.



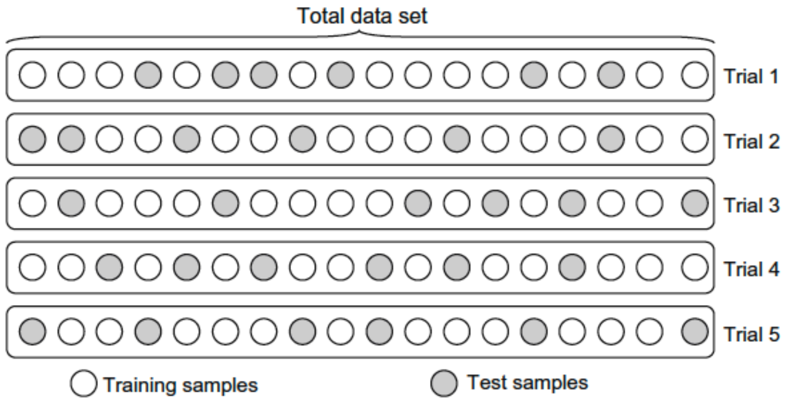


Fig. 5.34 Random subsampling cross-validation method

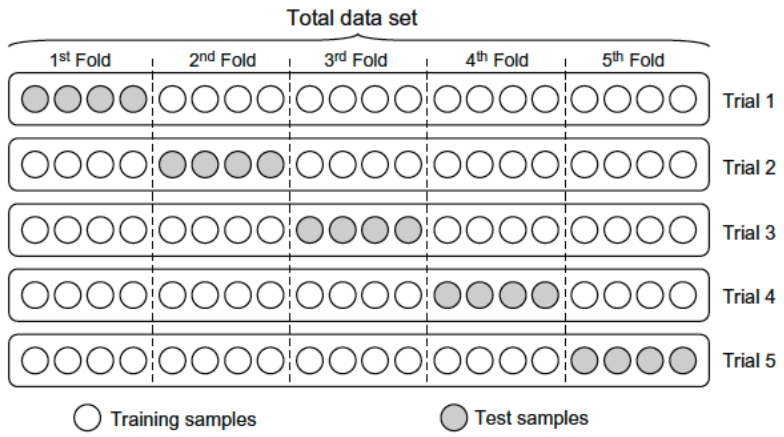


Fig. 5.35 *k*-fold cross-validation method

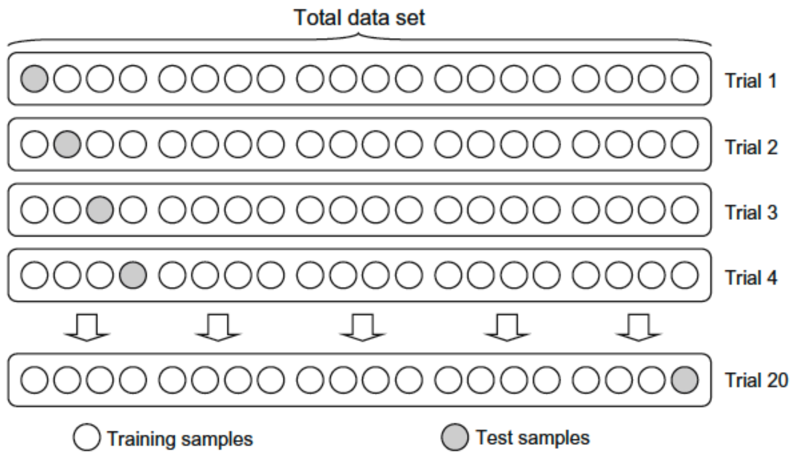
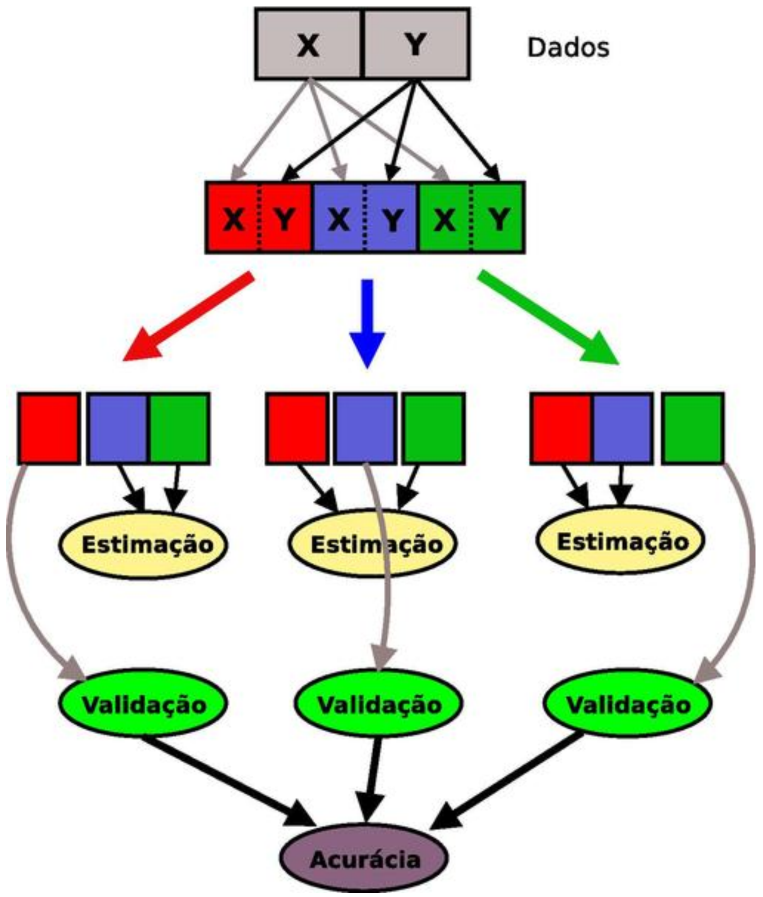


Fig. 5.36 Leave-one-out cross-validation method

K-fold

```
Begin {CROSS-VALIDATION algorithm}
    <1> Define the candidate topologies for the given problem;
    <2> Acquire the training and test subsets;
    <3> Apply the MLP learning algorithm to all candidate topologies
        using the training subsets;
    <4> Apply the test subset to the (already trained) candidate topologies
        in order to evaluate their generalization potential;
    <5> Obtain the final performance metric of each candidate topology
        with respect to the number of trials;
    <6> Select the candidate topology that obtained the best global
        performance;
    <7> If the global performance of the best candidate topology is within
        the precision required by the problem,
        <7.1> then: End the cross-validation process.
        <7.2> else: Specify a new set of candidate topologies and go
                    back to step <3>.
End {CROSS-VALIDATION algorithm}
```



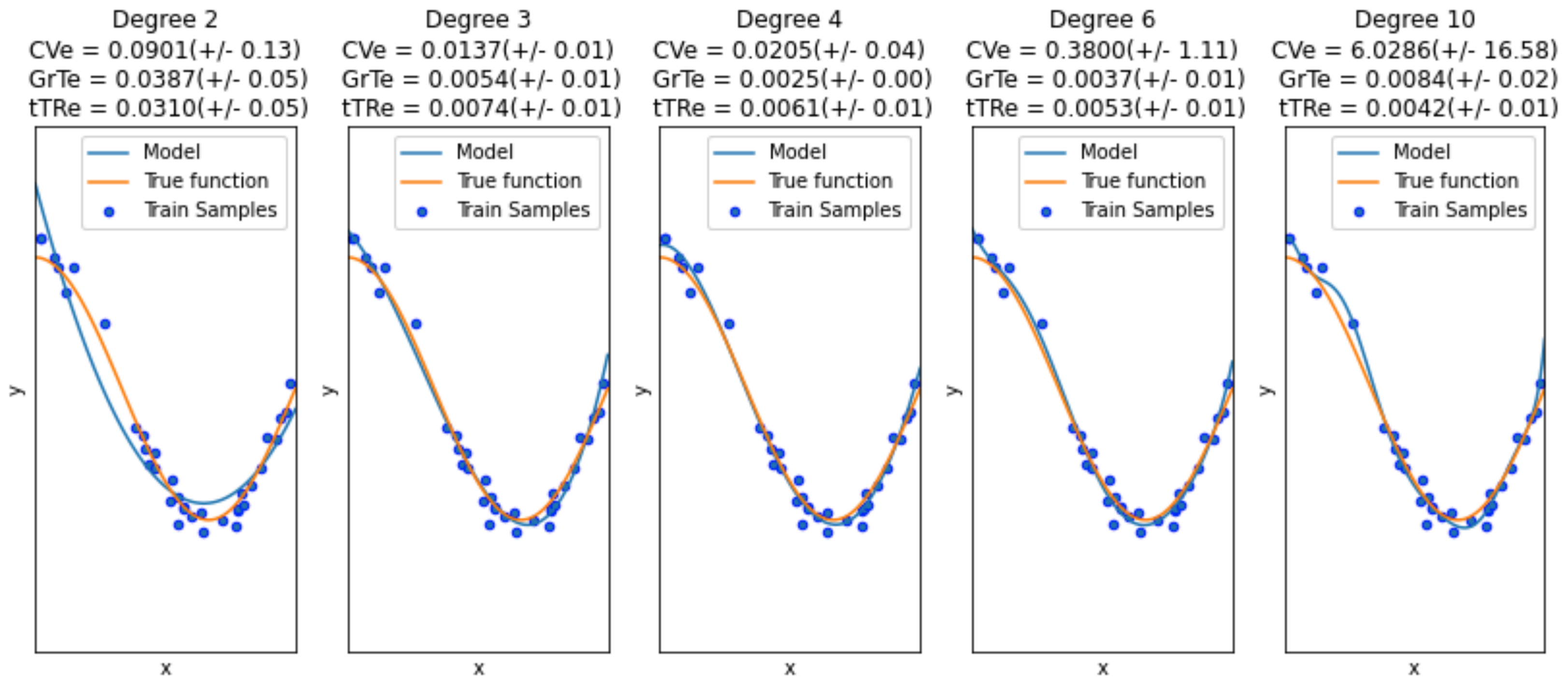
sklearn K-fold Polynomial Regression

```python
def true_fun(X):
    return np.cos(1.5 * np.pi * X)

np.random.seed(0)
n_samples = 35
degrees = [2, 3, 4, 6, 10]
X = np.sort(np.random.rand(n_samples))
y = true_fun(X) + np.random.randn(n_samples) * 0.1

# MSE errors: CVe-Cross Validation, GrTe-Ground Truth, tTRe, Training
```

```python
# Evaluate the models using crossvalidation
scores = cross_val_score(pipeline, X[:, np.newaxis], y,
                         scoring="neg_mean_squared_error", cv=10)
```


Degree 2
CVe = 0.0901(+/- 0.13)
GrTe = 0.0387(+/- 0.05)
tTRe = 0.0310(+/- 0.05)
Degree 3
CVe = 0.0137(+/- 0.01)
GrTe = 0.0054(+/- 0.01)
tTRe = 0.0074(+/- 0.01)
Degree 4
CVe = 0.0205(+/- 0.04)
GrTe = 0.0025(+/- 0.00)
tTRe = 0.0061(+/- 0.01)
Degree 6
CVe = 0.3800(+/- 1.11)
GrTe = 0.0037(+/- 0.01)
tTRe = 0.0053(+/- 0.01)
Degree 10
CVe = 6.0286(+/- 16.58)
GrTe = 0.0084(+/- 0.02)
tTRe = 0.0042(+/- 0.01)
Model
True function
Train Samples
y
x