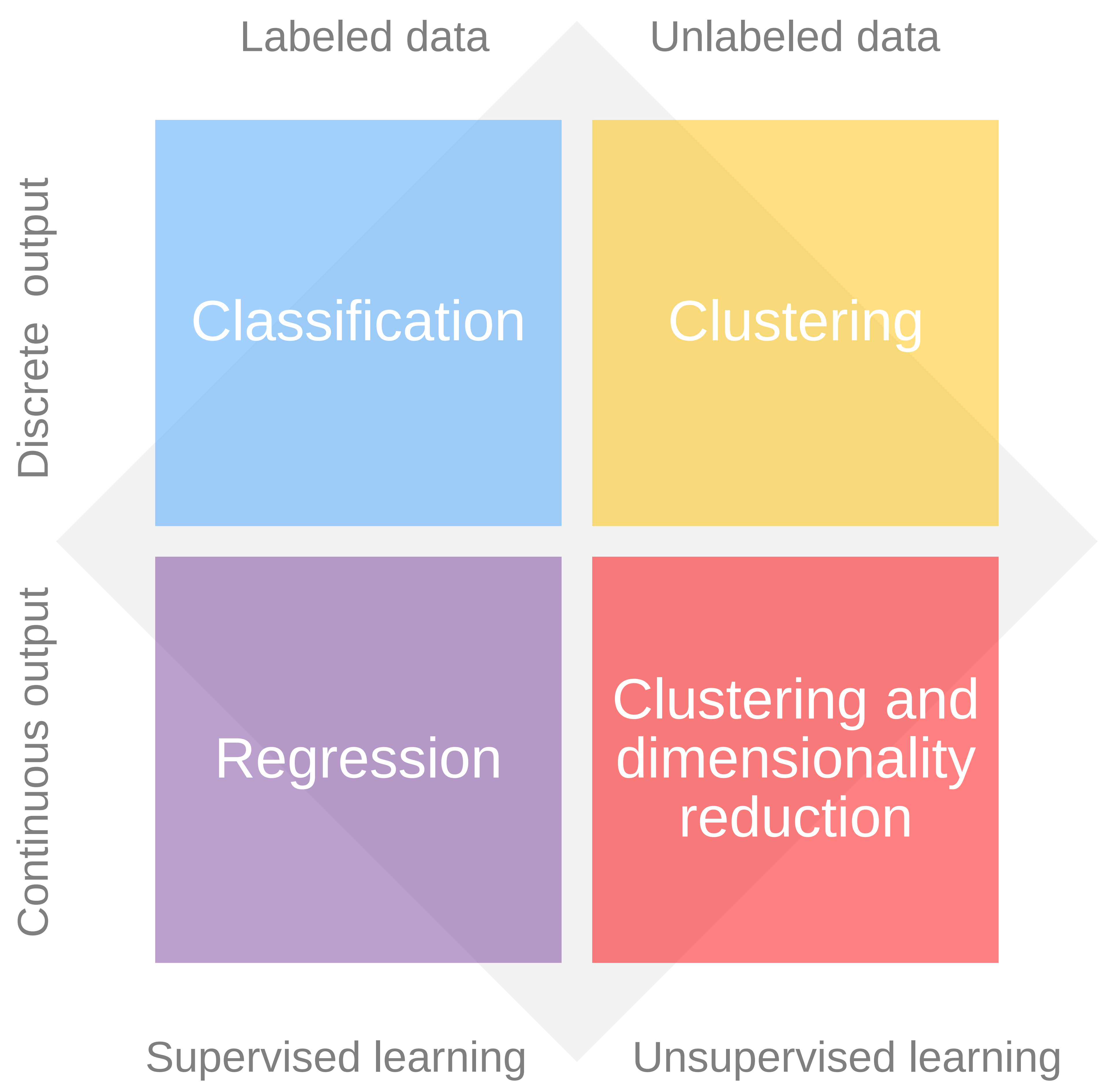
CREATIVE DATA MINING

Supervised Learning Algorithms

30.10.2017

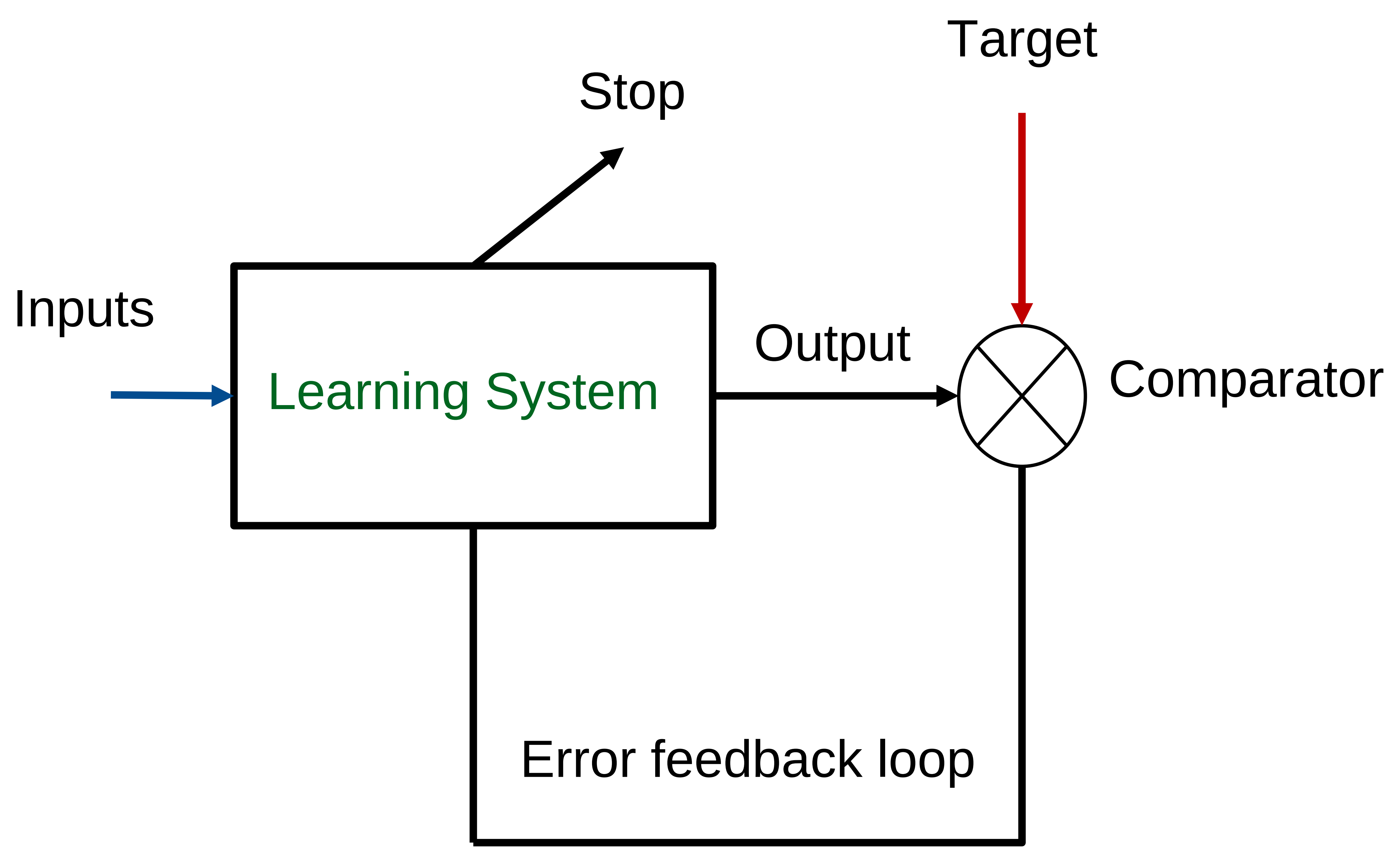
Dr. Varun OJHA

Danielle GRIEGO



ETH zürich
iA Chair of Information Architecture

Supervised Learning Systems



Terms to remember:

Input

Target (Known output)

Output (System’s output)

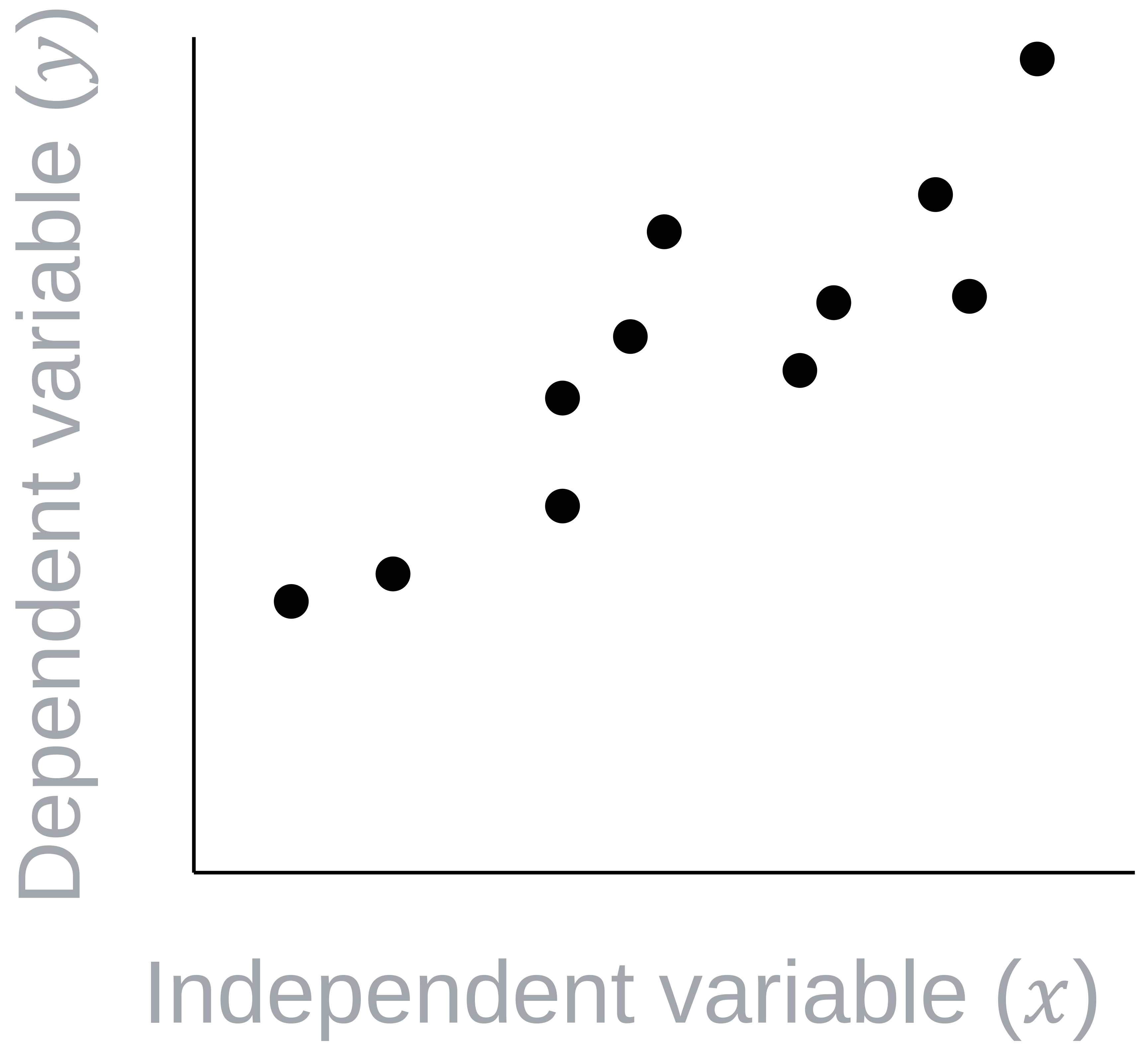
Feedback Loop (Training iteration)

Learning System (Model)

Stop (When to stop learning)

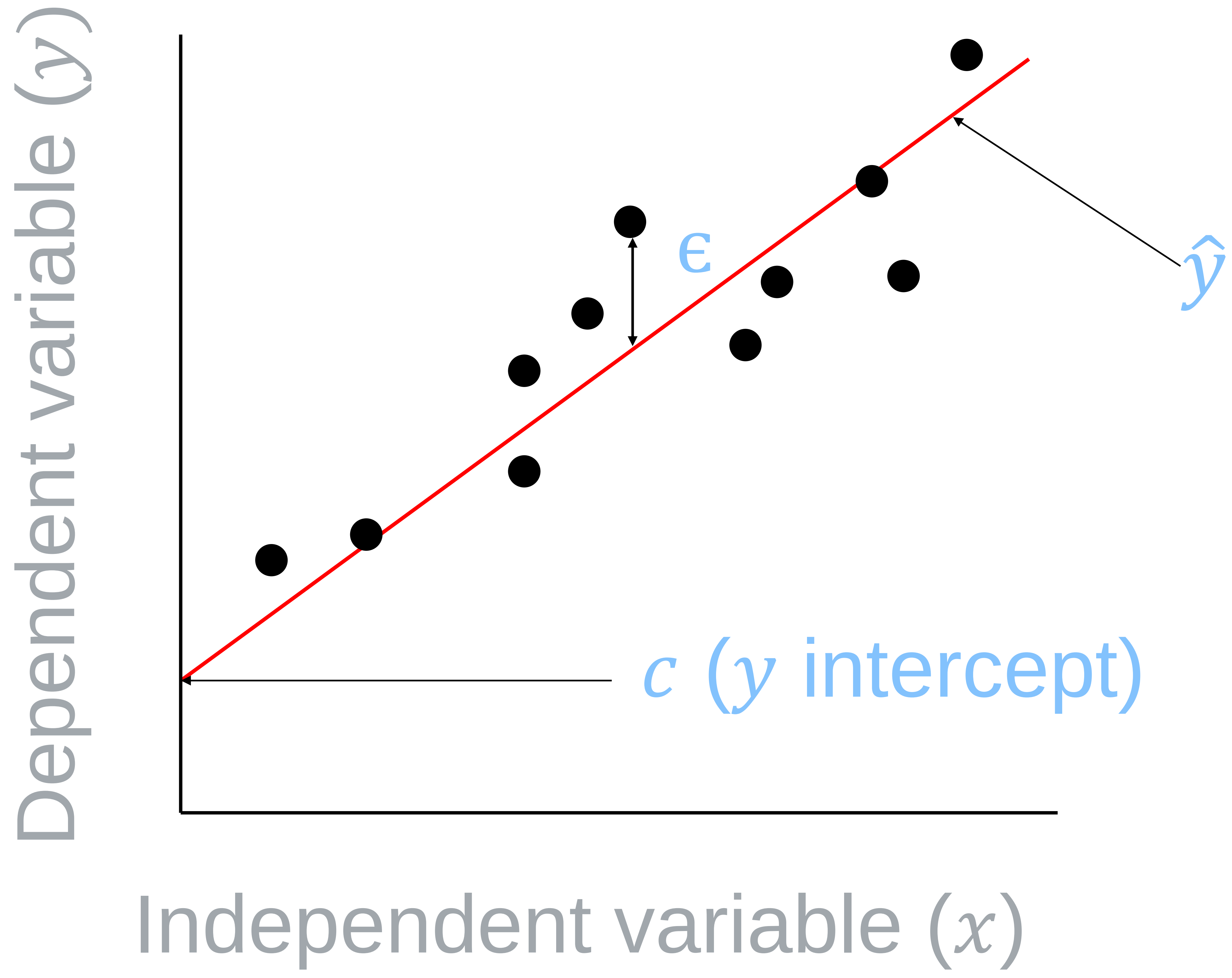
Supervised Learning Systems: Performance measures

Prediction error



Supervised Learning Systems: Performance measures

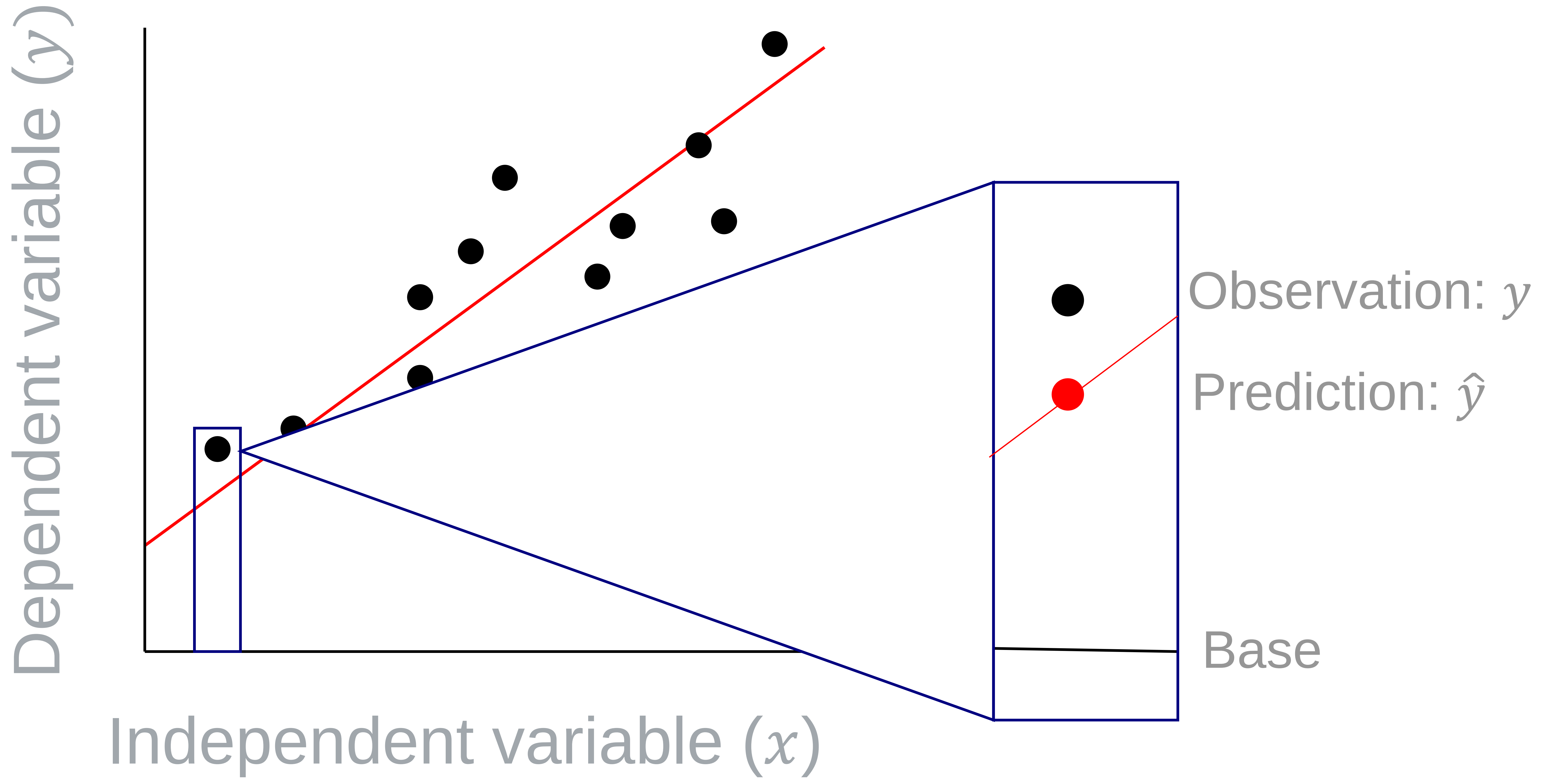
Prediction error



$m = \Delta y / \Delta x$ (slope)

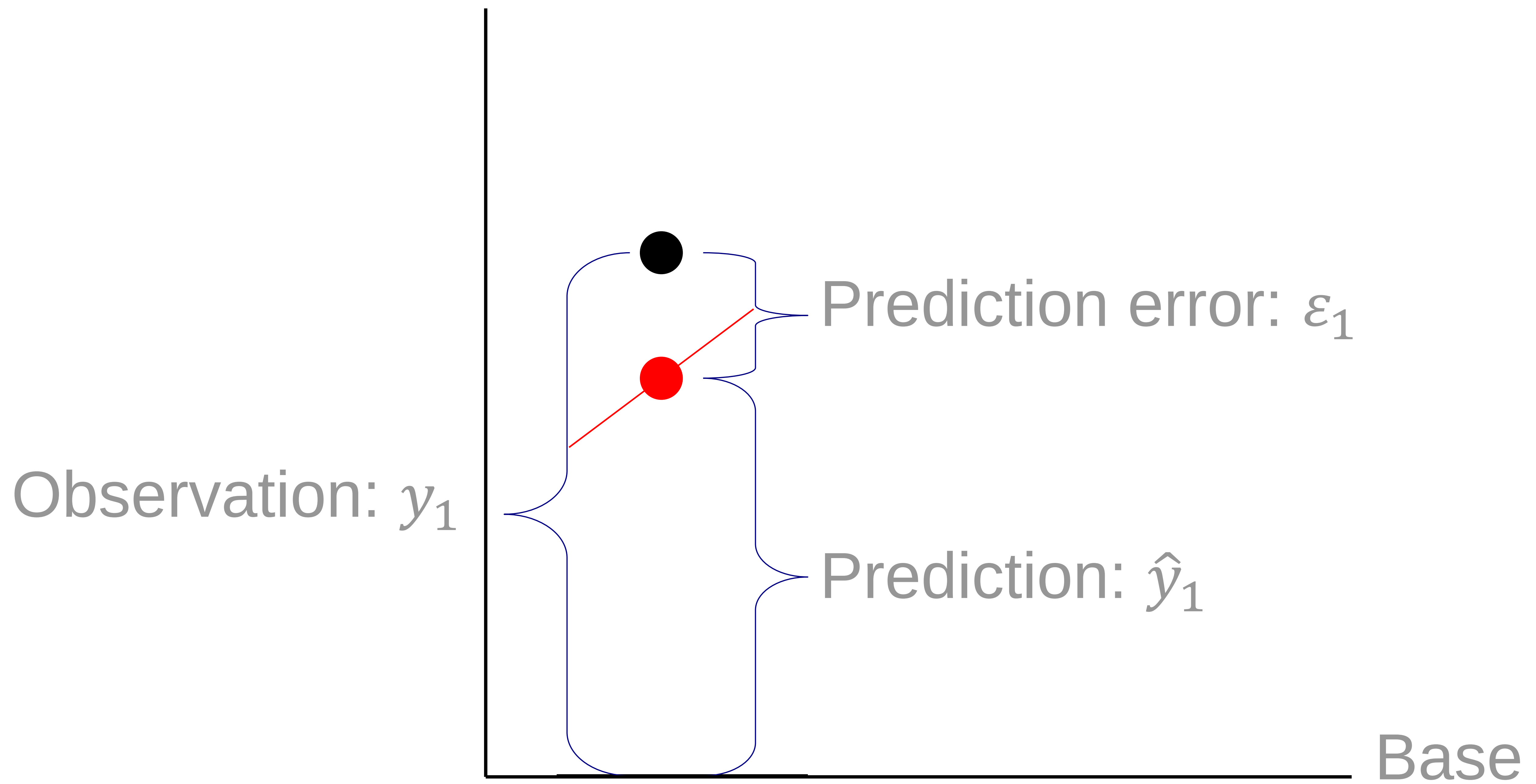
Supervised Learning Systems: Performance measures

Prediction error



Supervised Learning Systems: Performance measures

Prediction error



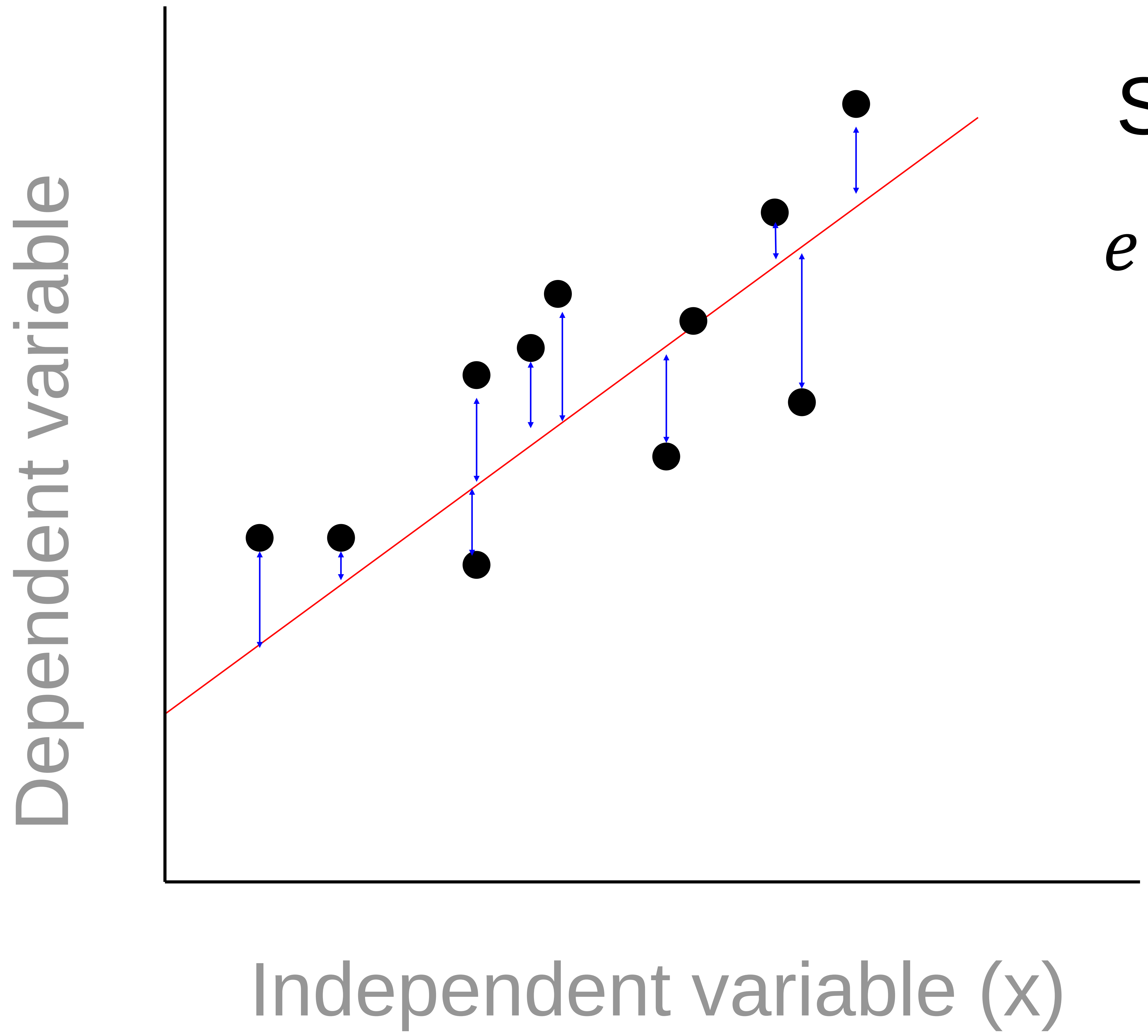
For each observation, the variation can be described as:

$$y_1 = \hat{y}_1 + \varepsilon_1$$

Actual = Prediction + Error

Supervised Learning Systems: Performance measures

Prediction error



Sum of Squares of Error (e)

$$e(t) = (\varepsilon_1)^2 + (\varepsilon_2)^2 + \cdots + (\varepsilon_n)^2$$

t is iteration number 1

Regression Problem

A Classical Example

Daily Electricity Energy Consumption (Average price)

Input Hydroelectric real [27881.8, 206035.0]

Input Nuclear real [114760.0, 187105.0]

Input Coal real [33537.0, 234833.0]

Input Fuel real [0.0, 67986.5]

Input Gas real [0.0, 84452.2]

Input Special real [5307.0, 16357.0]

X

Output Consume real **[0.765853, 5.11875]**

Y

365 Samples

Outputs are continues numbers between 0.7658 to 5.11875

Regression Problem

A Classical Example

Daily Electricity Energy Consumption (Average price)

5 Inputs

1 Output

365 Samples

MLP network (chosen)

5 inputs nodes

100 hidden nodes

1 output node

Common python import

```
import sys
import os

import csv
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
import numpy as np
```

Python code:

```
#Regression
electricity_consumption_data = pd.read_csv("dee.csv") # fetching entire data
x = electricity_consumption_data.iloc[:,0:6] # selecting inputs
inputs = x.as_matrix() # converting inputs as matric from DataFrame
y =  electricity_consumption_data.iloc[:,6:7] # selecting output
target = y.as_matrix() # converting inputs as matric from DataFrame

#Splitting training data and test data apart
X_train, X_test, y_train, y_test = train_test_split(inputs, target, test_size=0.33, random_state=42)

#Preprocessing :
from random import randint
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler() #  Scalar transformations of the data
scaler.fit(X_train) # Fit only to the training data

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

#Constructing MLP
from sklearn.neural_network import  MLPRegressor
mlpreg = MLPRegressor(solver='sgd', alpha=1e-5, hidden_layer_sizes=(10), random_state=1)
mlpreg.fit(X_train, y_train)  # training MLP
y_pred = mlpreg.predict(X_test) #prediction
# Evaluating performance of Constructing MLP
plt.scatter(y_test, y_pred,  color='black')
from sklearn.metrics import mean_squared_error
print("Mean Squared Error:",mean_squared_error(y_test, predicted))
from sklearn.metrics import r2_score
print("R2 (Squared Correaltion-coeff):",r2_score(y_test, y_pred))
```

Classification Problem

A Classical Example

Iris flower identification

Input Sepal Length real[4.3,7.9]

Input Sepal Width real[2.0,4.4] X

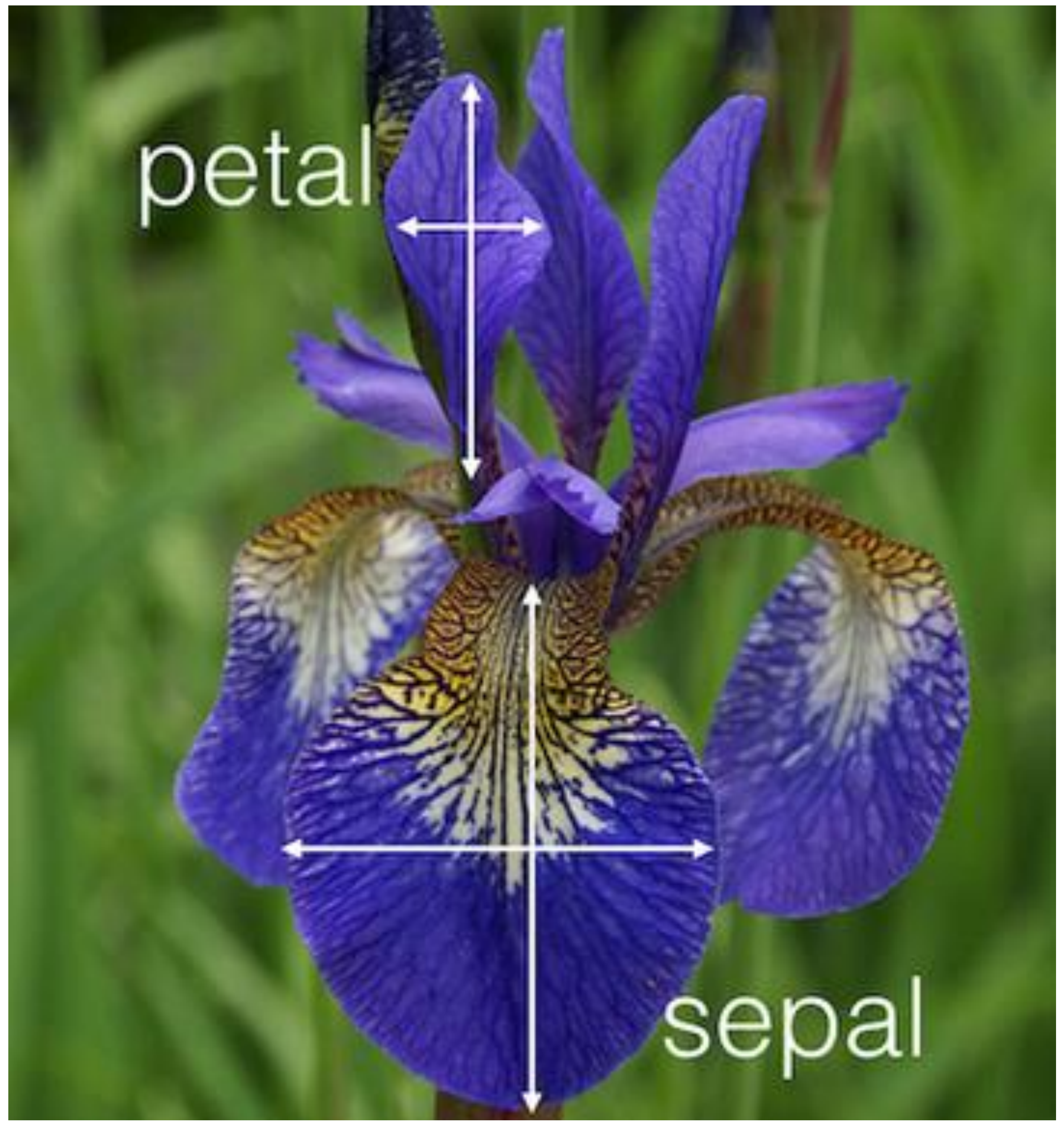
Input Petal Length real[1.0,6.9]

Input Petal Width real[0.1,2.5]

Output {Iris-Setosa, Iris-Versicolor, Iris-Virginica} Y

150 Samples

Outputs are discrete



Setosa

Versicolor

Virginica

Classification Problem

A Classical Example

Iris flower identification

4 Inputs

3 Output

150 Samples

MLP network (chosen)

4 inputs nodes

10 hidden nodes

3 output node

Common python imports

```
import sys
import os

import csv
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
import numpy as np
```

Python code:

```
#Classification
irisdata = pd.read_csv("iris.csv", header=None) # Fetching data
irisdata.columns = ['SL', 'SW','PL','PW','Flower_Class'] # Remaining columns
x = irisdata[['SL', 'SW','PL','PW']]  # fetching inputs
inputs = x.as_matrix() # Converting Data Frame to matrix format
y = irisdata[['Flower_Class']] # fetching outputs
target = y.as_matrix()  # Converting Data Frame to matrix format

#Splitting training data
X_train, X_test, y_train, y_test = train_test_split(inputs, target, test_size=0.33, random_state=42)
# Preprocessing
scaler = StandardScaler()
scaler.fit(X_train) # Fit only to the training data
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

# MLP Construction
from sklearn.neural_network import MLPClassifier
mlpclass = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5), random_state=1)
mlpclass.fit(X_train, y_train) #MLP TRAINING

# Evaluating performance of Constructing MLP
y_pred = mlpclass.predict(X_test)
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_pred)*100,"%")
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Thank you!

Varun Ojha ojha@arch.ethz.ch

Danielle Griego griego@arch.ethz.ch