CS 1675 Introduction to ML
Lecture 3

Designing a learning system

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square, x4-8845

people.cs.pitt.edu/~milos/courses/cs1675/

Homework assignment

Homework assignment 1 will be out today

Two parts: **Report + Programs**

Submission:

- via Courseweb
- Report (submit in pdf)
- Programs (submit using a zip or tar archive file)
- Deadline 11:00am on September 14, 2017 (prior to the lecture)

Rules:

- Strict deadline
- No collaboration policy, reports and programs must be done individually

Learning: first look

1. Data: $D = \{d_1, d_2, .., d_n\}$

2. Model selection:

– **Select a model** or a set of models (with parameters)

E.g. $y = ax + b$

3. Choose the objective function

– **Squared error** $\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$

4. Learning:

- **Find the set of parameters optimizing the error function**

– The model and parameters with the smallest error

5. Application

– **Apply the learned model to new data**

– E.g. predict ys for new inputs **x** using learned $f(\mathbf{x})$

Learning: first look

1. Data: $D = \{d_1, d_2, .., d_n\}$

2. Model selecti

– **Selec**

E.g.

3. Choose the

– **Squared e**

4. Learning:

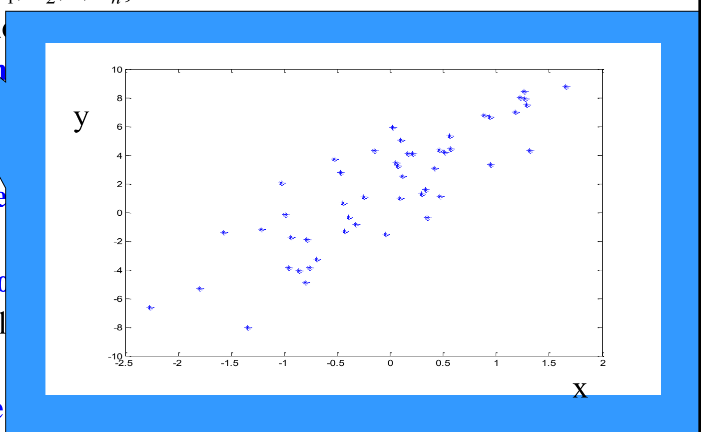
- **Find the set o**

– The model

5. Application:

– **Apply the**

– E.g. predict ys for new inputs **x** using learned $f(\mathbf{x})$



A learning system: basic cycle

1. Data: $D = \{d_1, d_2, .., d_n\}$

2. Model selection:

– **Select a model** or a set of models (with parameters)

E.g. $y = ax + b$

3. Choose the

– **Squa**

4. Learning:

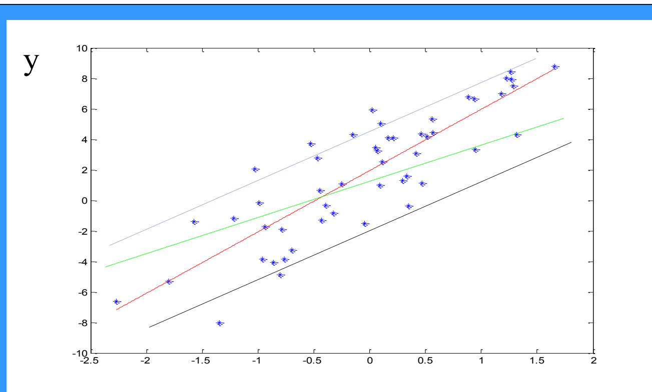
• **Find the se**

– The mod

5. Applicatio

– **Apply t**

– E.g. pred



Learning: first look

1. Data: $D = \{d_1, d_2, .., d_n\}$

2. Model selection:

– **Select a model** or a set of models (with parameters)

E.g. $y = ax + b$

3. Choose the objective function $\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$

– **Squared error**

4. Learning:

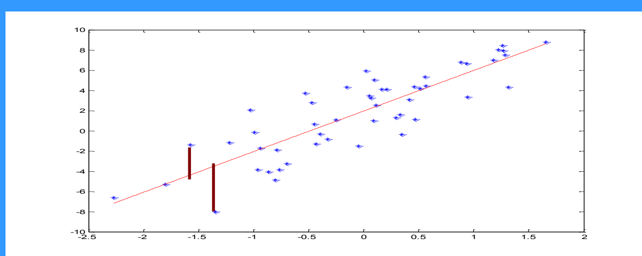
• **Find the se**

– The mod

5. Applicatio

– **Apply t**

– E.g. pred



Learning: first look

1. Data: $D =$

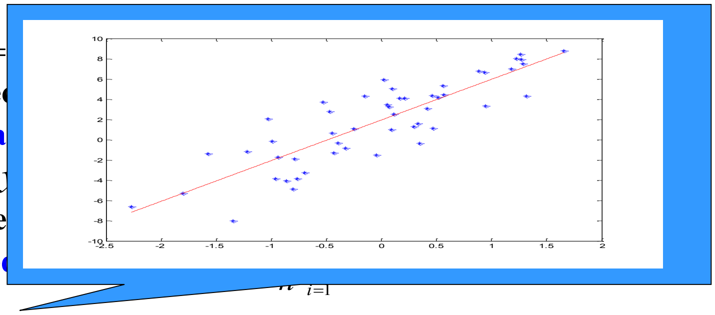
2. Model selec

– **Select a**

E.g. y

3. Choose the

– **Square**



$n \quad i=1$

4. Learning:

- **Find the set of parameters optimizing the error function**
  – The model and parameters with the smallest error

5. Application:

– **Apply the learned model to new data**

– E.g. predict ys for new inputs $\mathbf{x}$ using learned $f(\mathbf{x})$

Learning: first look

1. Data: $D = \{d_1, d_2, .., d_n\}$

2. Model selection:

– **Select a model** or a set of models (with parameters)

E.g.

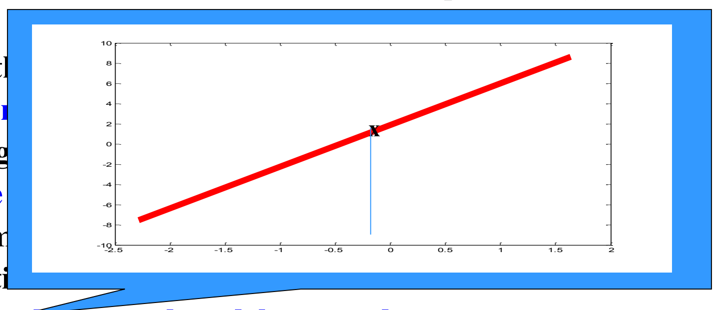
3. Choose t

– **Squar**

4. Learning

- **Find the**
  – The m

5. Applicati



– **Apply the learned model to new data**

– E.g. predict ys for new inputs $\mathbf{x}$ using learned $f(\mathbf{x})$

Learning: first look

1. Data: $D = \{d_1, d_2, .., d_n\}$

2. Model selection:

– **Select a model** or a set of models (with parameters)

E.g. $y = ax + b$

3. Choose the objective function

– **Squared error** $\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$

4. Learning:

- **Find the set of parameters optimizing the error function**
  – The model and parameters with the smallest error

5. Application

– **Apply the learned model to new data**

- **Looks straightforward, but there are problems ….**

Learning: generalization error

We fit the model based on past examples observed in ***D***

Training data: **Data used to fit the parameters of the model**

Training error:

$$Error(D, f) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$$

Problem: Ultimately we are interested in learning the mapping that performs well on the whole population of examples

True (generalization) error (over the whole population):

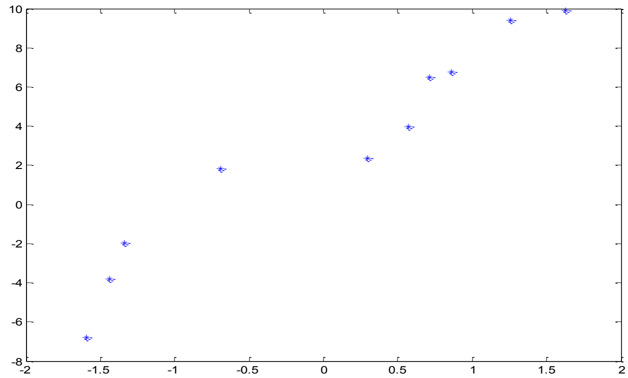
$$E_{(x,y)}[(y - f(x))^2] \quad \text{Mean squared error}$$

Training error tries to approximate the true error !!!!

Does a good training error imply a good generalization error ?

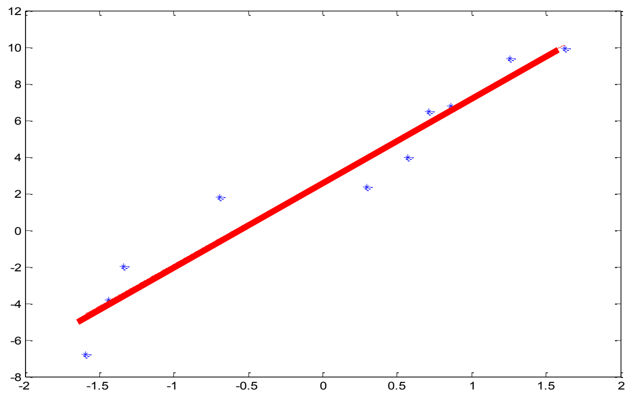
Overfitting

- Assume we have a set of 10 points and we consider polynomial functions as our possible models



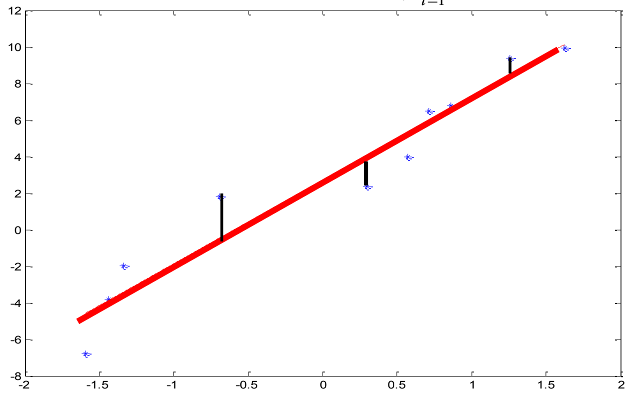
Overfitting

- Fitting a linear function with the square error
- **Error is nonzero. Why?**



Overfitting

- Fitting a linear function with the square error
- **Error is nonzero:** $Error(D,f) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$

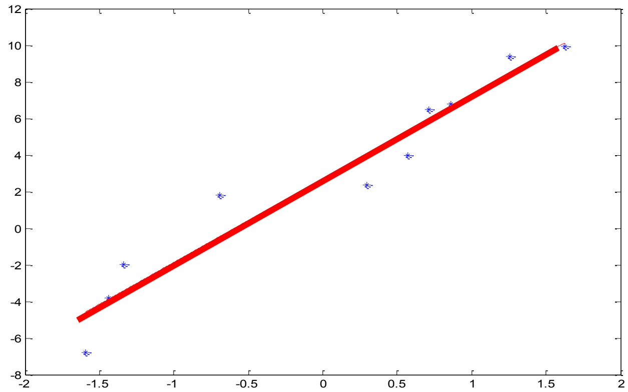


Overfitting

Assume in addition to linear model: $y = f(x) = ax + b$

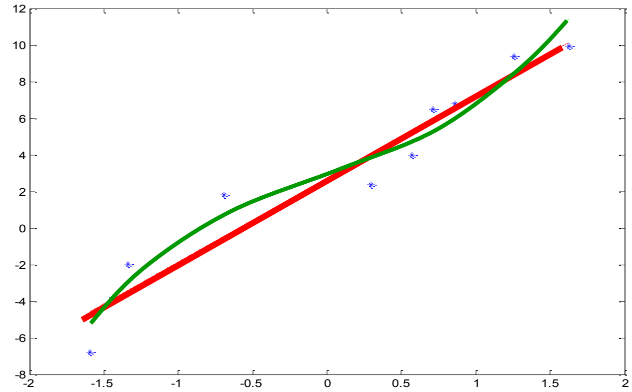
we consider also: $y = f(x) = a_3x^3 + a_2x^2 + a_1x + b$

Which model would give us a smaller error for the least squares fit?



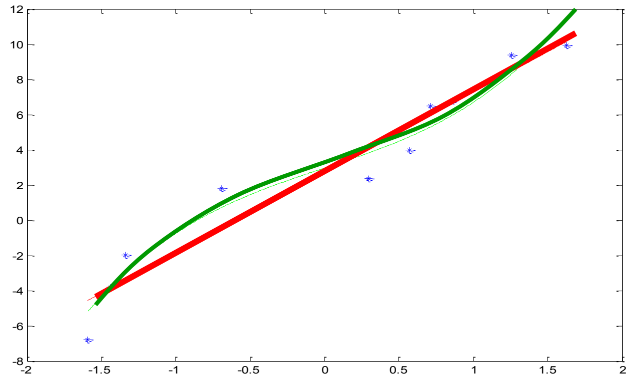
Overfitting

- Linear vs. cubic polynomial
- Higher order polynomial leads to a better fit, smaller error



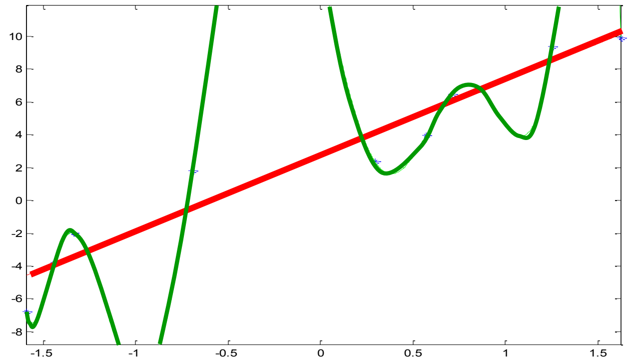
Overfitting

- Is it always good to minimize the error of the observed data?



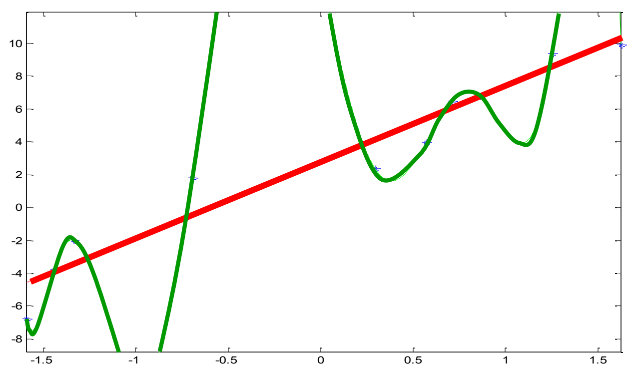
Overfitting

- For 10 data points, the degree 9 polynomial gives a perfect fit (Lagrange interpolation). Error is zero.
- Is it always good to minimize the training error?



Overfitting

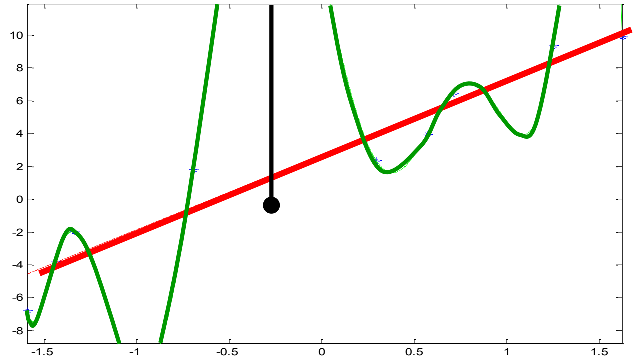
- For 10 data points, degree 9 polynomial gives a perfect fit (Lagrange interpolation). Error is zero.
- Is it always good to minimize the training error? NO !!
- **More important:** How do we perform on the unseen data?



Overfitting

Situation when the training error is low and the generalization error is high. Causes of the phenomenon:

- Model with a large number of parameters (degrees of freedom)
- Small data size (as compared to the complexity of the model)



How to evaluate the learner's performance?

- **Generalization error** is the true error for the population of examples we would like to optimize

$$E_{(x,y)}[(y - f(x))^2]$$

- But it cannot be computed exactly
- **Sample mean only approximates the true mean**

- **Optimizing the training error can lead to the overfit, i.e.** training error may not reflect properly the generalization error

$$\frac{1}{n}\sum_{i=1,..n}(y_i - f(x_i))^2$$

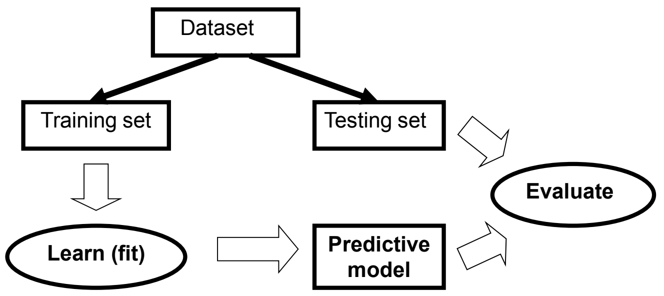
- So how to assess the generalization error?

How to evaluate the learner's performance?

- **Generalization error** is the true error for the population of examples we would like to optimize
- **Sample mean only approximates it**
- **Two ways to assess the generalization error is:**
  - **Theoretical: Law of Large numbers**
    - statistical bounds on the difference between true and sample mean errors
  - **Practical:** Use a separate data set with m data samples to test the model
    - **(Average) test error** $\frac{1}{m}\sum_{j=1,..m}(y_j - f(x_j))^2$

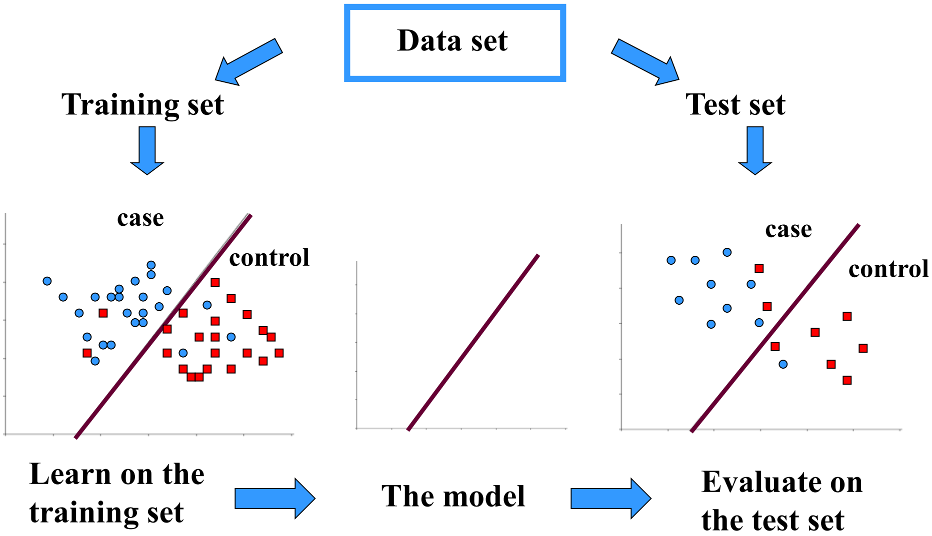
Testing of learning models

- **Simple holdout method**
  - Divide the data to the training and test data



  - Typically 2/3 training and 1/3 testing

Testing of models



Evaluation measures

Regression:

- Squared error
- Absolute error
- Mean absolute percentage error

Classification:

		Actual: Case	Actual: Control
Prediction	Case	TP 0.3	FP 0.1
	Control	FN 0.2	TN 0.4

Misclassification error:

$$E = FP + FN$$

Sensitivity:

$$SN = \frac{TP}{TP + FN}$$

Specificity:

$$SP = \frac{TN}{TN + FP}$$

A learning system: basic cycle

1. Data: $D = \{d_1, d_2, .., d_n\}$

2. Model selection:

– **Select a model** or a set of models (with parameters)

E.g. $y = ax + b$

3. Choose the objective function

– **Squared error** $\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$

4. Learning:

- **Find the set of parameters optimizing the error function**

– The model and parameters with the smallest error

5. Testing/validation:

– **Evaluate on the test data**

6. Application

– **Apply the learned model to new data** $f(\mathbf{x})$

A learning system: basic cycle

1. Data: $D = \{d_1, d_2, .., d_n\}$

2. Model selection:

– **Select**

E.g.

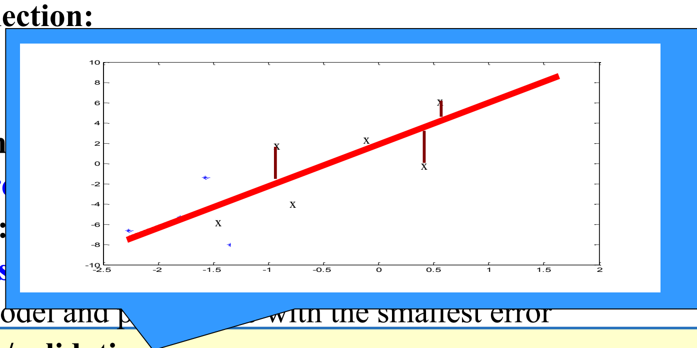
3. Choose th

– **Square**

4. Learning:

- **Find the s**

– The model and p with the smallest error



5. 5. Testing/validation:

– **Evaluate on the test data**

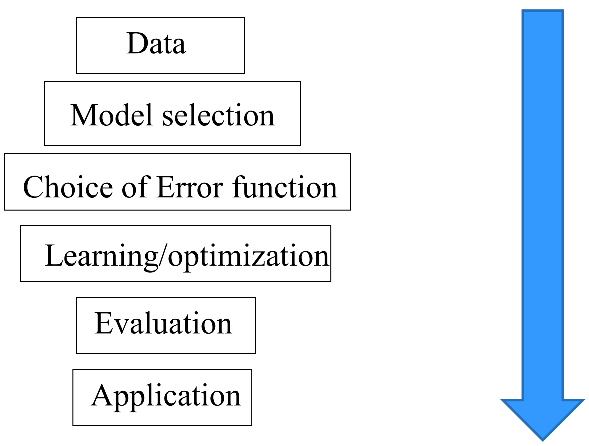
6. Application

– **Apply the learned model to new data** $f(\mathbf{x})$

A learning system: basic cycle

1. **Data:** $D = \{d_1, d_2, .., d_n\}$
2. **Model selection:**
   – **Select a model** or a set of models (with parameters)
     E.g. $y = ax + b$
3. **Choose the objective function**
   – **Squared error** $\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$
4. **Learning:**
- **Find the set of parameters optimizing the error function**
  – The model and parameters with the smallest error
5. **Testing/validation:**
   – **Evaluate on the test data**
6. **Application**
   – **Apply the learned model to new data** $f(\mathbf{x})$

Steps taken when designing an ML system



Add some complexity

Data

Data cleaning/preprocessing

Feature selection/dimensionality reduction

Model selection

Choice of Error function

Learning/optimization

Evaluation

Application

Designing an ML solution

Data

Data cleaning/preprocessing

Feature selection/dimensionality reduction

Model selection

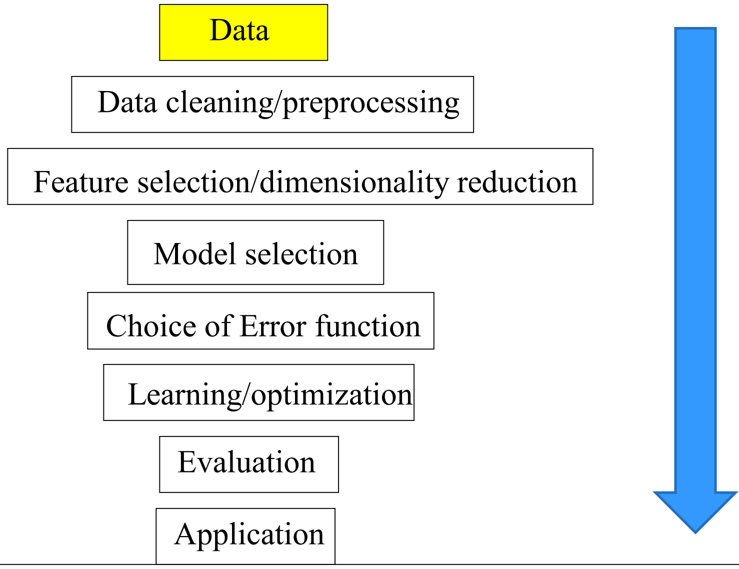
Choice of Error function

Learning/optimization

Evaluation

Application

Designing an ML solution



Data source and data biases

- **Understand the data source**
- **Understand the data your models will be applied to**
- **Watch out for data biases:**
  - Make sure the data we make conclusions on are the same as data we used in the analysis
  - It is very easy to derive “unexpected” results when data used for analysis and learning are biased

- **Results (conclusions) derived for a biased dataset do not hold in general !!!**

Data biases

Example: Assume you want to build an ML program for predicting the stock behavior and for choosing your investment strategy

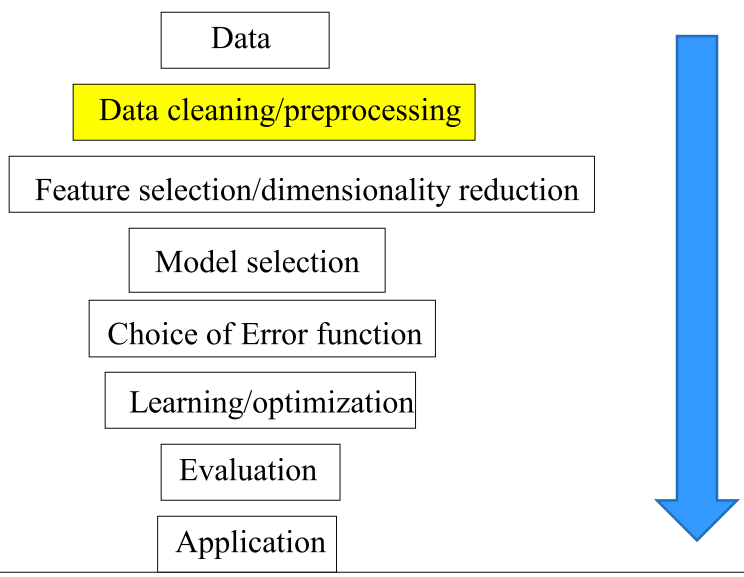
Data extraction:

- pick companies that are traded on the stock market on January 2017
- Go back 30 years and extract all the data for these companies
- Use the data to build an ML model supporting your future investments

Question:

- **Would you trust the model?**
- **Are there any biases in the data?**

Steps taken when designing an ML system



Data cleaning and preprocessing

Data you receive may not be perfect:

- **Cleaning**
- **Preprocessing (conversions)**

Cleaning:

- **Get rid of errors, noise,**
- **Removal of redundancies**

Preprocessing:

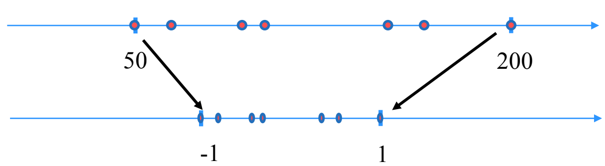
- **Renaming**
- **Rescaling (normalization)**
- **Discretizations**
- **Abstraction**
- **Aggregation**
- **New attributes**

Data preprocessing

- **Renaming** (relabeling) categorical values to numbers
  - dangerous in conjunction with some learning methods
  - numbers will impose an order that is not warranted

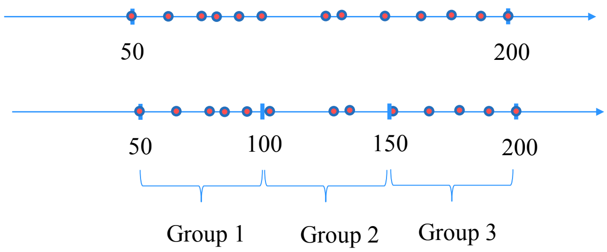
High → 2 Normal → 1 Low → 0	True → 2 False → 1 Unknown → 0	Red → 2 Blue → 1 Green → 0

- **Rescaling (normalization):** continuous values transformed to some range, typically [-1, 1] or [0,1].

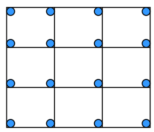


Data preprocessing

- **Discretizations (binning):** continuous values to a finite set of discrete values
- **Example:**



- **Another Example:**



Data preprocessing

- **Abstraction:** merge together categorical values
- **Aggregation:** summary or aggregation operations, such minimum value, maximum value, average etc.
- **New attributes:**
  - example: obesity-factor = weight/height