CS 1675 Intro to Machine Learning

Lecture 2

Designing a learning system

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

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

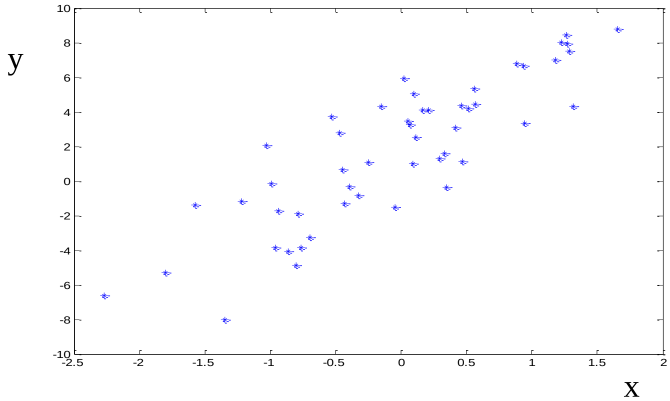
Administrivia

- **No homework assignment this week**

- Please try to obtain a copy of Matlab:
  http://technology.pitt.edu/software/matlab-students
- **Next week:**
  - **Recitations:** Matlab tutorial
  - **Tuesday:** Review of algebra and probability

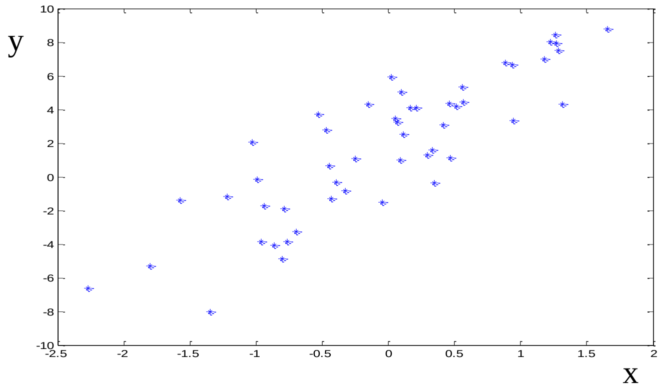
Learning: first look

- Assume we see examples of pairs ($\mathbf{x}$, y) in D and we want to learn the mapping $f : X \rightarrow Y$ to predict y for some future **x**
- We get the data D - what should we do?



Learning: first look

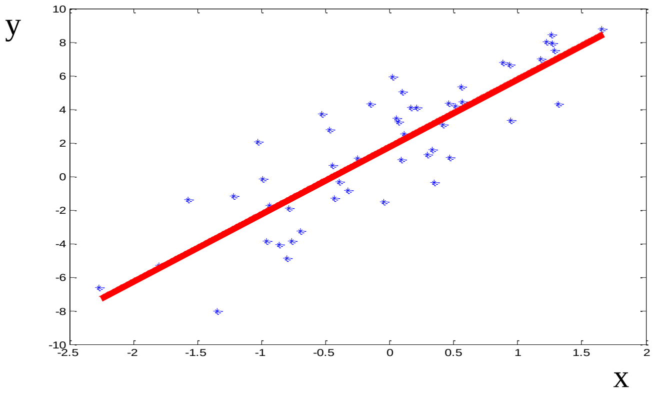
- **Problem:** many possible functions $f : X \rightarrow Y$ exists for representing the mapping between **x** and y
- Which one to choose? Many examples still unseen!



Learning: first look

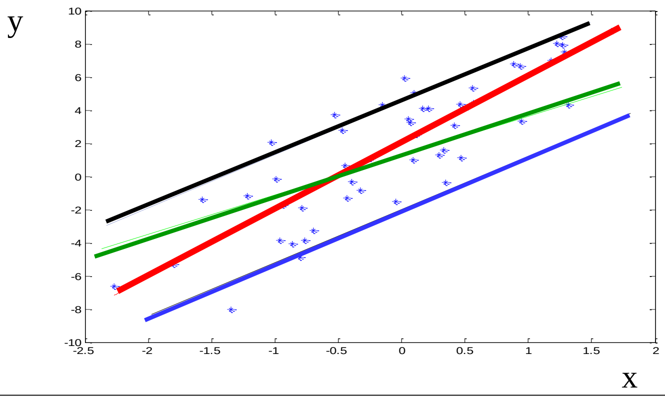
- **Solution:** make an assumption about the model, say,

$$f(x) = ax + b$$



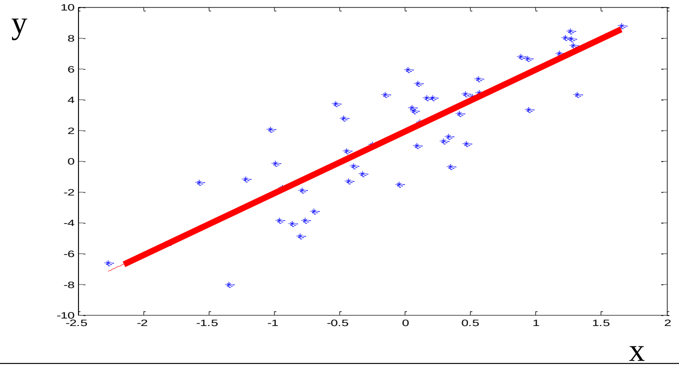
Learning: first look

- Choosing a parametric model or a set of models is not enough
  Still too many functions $f(x) = ax + b$
  - One for every pair of parameters *a*, *b*



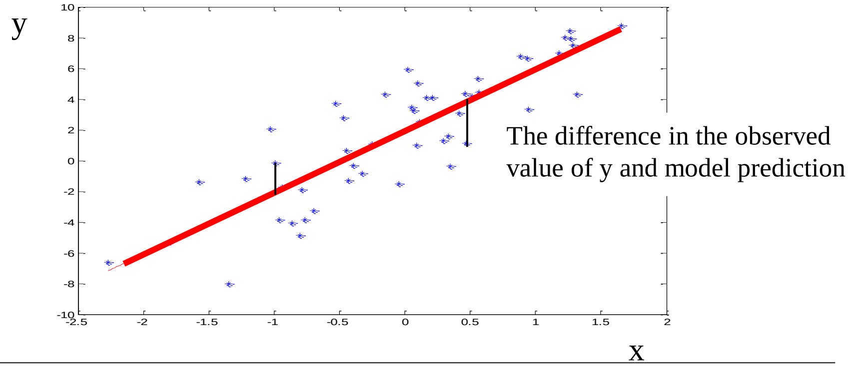
Learning: first look

- We want the **best set** of model parameters
  - reduce the misfit between the model **M** and observed data ***D***
  - Or, (in other words) explain the data the best
- **How to measure the misfit?**



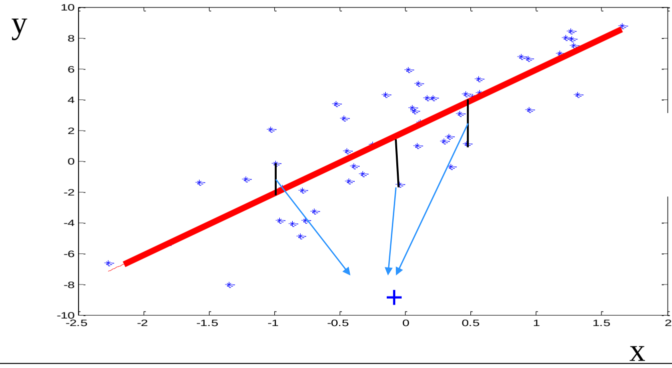
Learning: first look

- We want the **best set** of model parameters
  - reduce the misfit between the model **M** and observed data ***D***
  - Or, (in other words) explain the data the best
- **How to measure the misfit?**



Learning: first look

- We want the **best set** of model parameters
  - reduce the misfit between the model **M** and observed data ***D***
  - Or, (in other words) explain the data the best
- **How to measure the misfit?**



Learning: first look

- We want the **best set** of model parameters
  - reduce the misfit between the model **M** and observed data ***D***
  - Or, (in other words) explain the data the best
- **How to measure the misfit?**

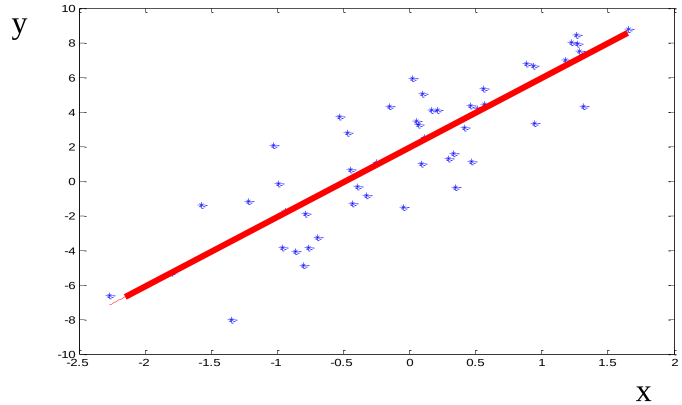
Objective function:

- **Error (loss) function: Measures the misfit between *D* and M**
- **Examples of error functions:**
  - Average Square Error $$\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$$
  - Average Absolute Error $$\frac{1}{n}\sum_{i=1}^{n}|y_i - f(x_i)|$$

Learning: first look

- **Linear regression**
- Minimizes the squared error function for the linear model

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



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 (error) function

- **Squared error** $Error(D,a,b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - ax_i - b))^2$

4. Learning:

- **Find the set of parameters (*a*,*b*) optimizing the error function**

$$(a^*, b^*) = \arg\max_{(a,b)} Error(D,a,b)$$

5. Application

- **Apply the learned model to new data** $f(x) = a^* x + b^*$
- E.g. predict ys for the new input x

Learning: first look

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

2. **Model selecti**

– **Selec**

E.g.

3. Choose the

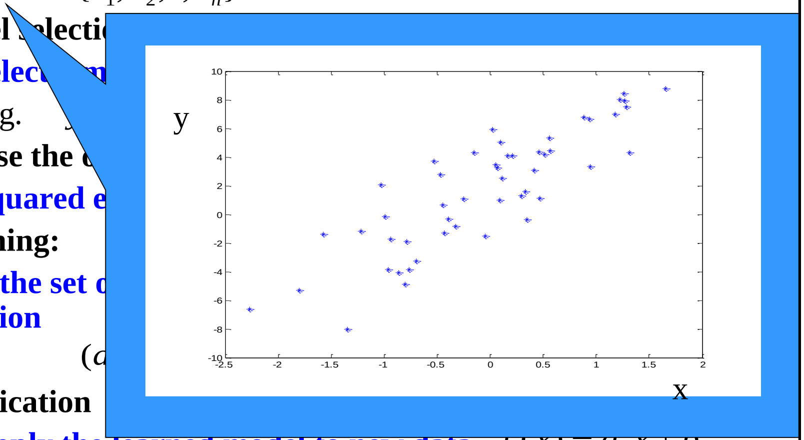
– **Squared e**

4. Learning:

- **Find the set o**
  function

$(a$

5. Application



– **Apply the learned model to new data** $f(x) = a^* x + b^*$

– E.g. predict ys for the new input 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

– **Squa**

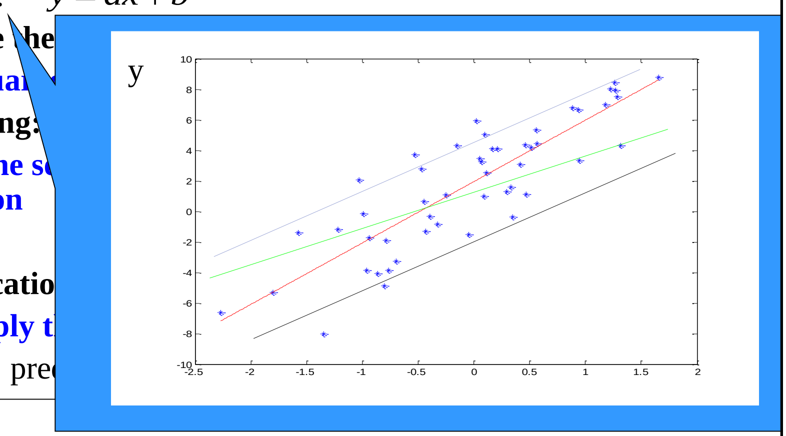
4. Learning:

- **Find the se**
  function

5. Applicatio

– **Apply t**

– E.g. pre

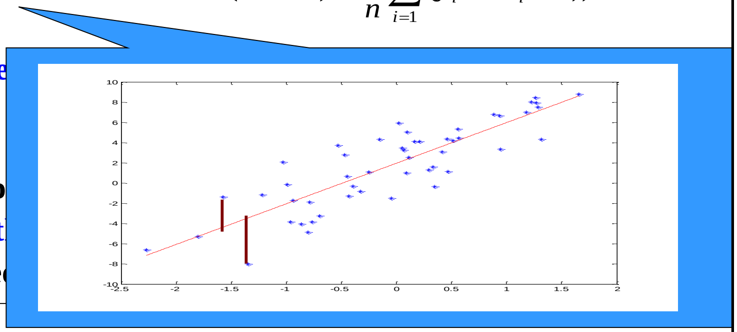


Learning: first look

1. **Data:** $D = \{d_1, d_2, .., d_n\}$
2. **Model selection:**
   – **Select a model** or a set of models (with parameters)

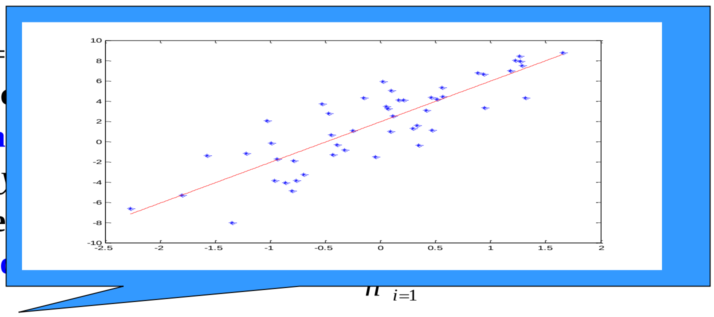
     E.g. $y = ax + b$

3. **Choose the objective (error) function**
   – **Squared error** $Error(D, a, b) = \frac{1}{n} \sum_{i=1}^{n} (y_i - ax_i - b))^2$
4. **Learning:**
   • **Find the se**
     function



5. **Applicatio**
   – **Apply t**
   – E.g. pre

Learning: first look



1. **Data:** $D =$
2. **Model sele**
   – **Select a**

     E.g. y

3. **Choose the**
   – **Square**

n $i=1$

4. **Learning:**
   • **Find the set of parameters (*a*,*b*) optimizing the error function**

$$(a^*, b^*) = \arg\max_{(a,b)} Error(D, a, b)$$

5. **Application**
   – **Apply the learned model to new data** $f(x) = a^* x + b^*$
   – E.g. predict ys for the new input 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)

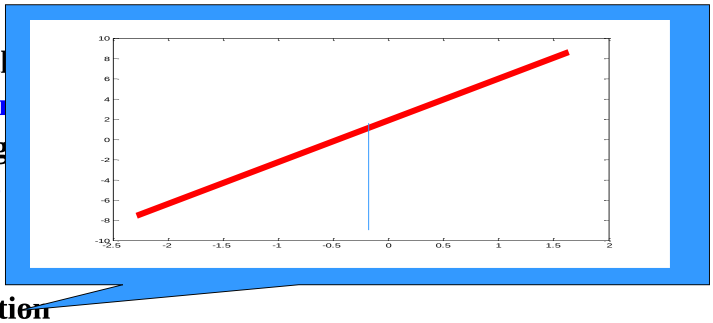
E.g.

3. Choose t

– **Squar**

4. Learning

- **Find the function**



5. Application

– **Apply the learned model to new data** $f(x) = a^* x + b^*$

– E.g. predict ys for the new input 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 (error) function

– **Squared error** $Error(D, a, b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - ax_i - b))^2$

4. Learning:

- **Find the set of parameters (*a*,*b*) optimizing the error function** $(a^*, b^*) = \arg\max_{(a,b)} Error(D, a, b)$

5. Application

– **Apply the learned model to new data** $f(x) = a^* x + b^*$

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:

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

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):

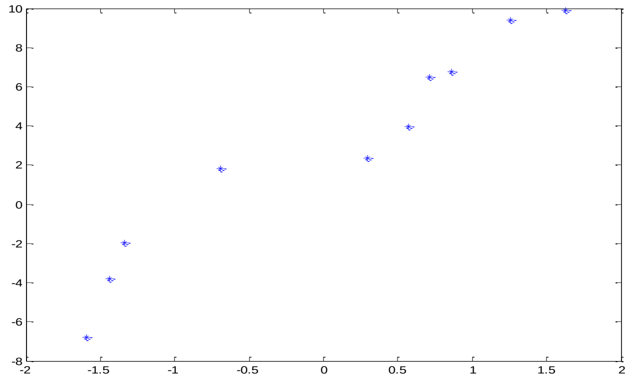
$$Error(a,b) = E_{(x,y)}[(y - f(x))^2]$$ Mean squared error

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

Does a good training error imply a good generalization error ?

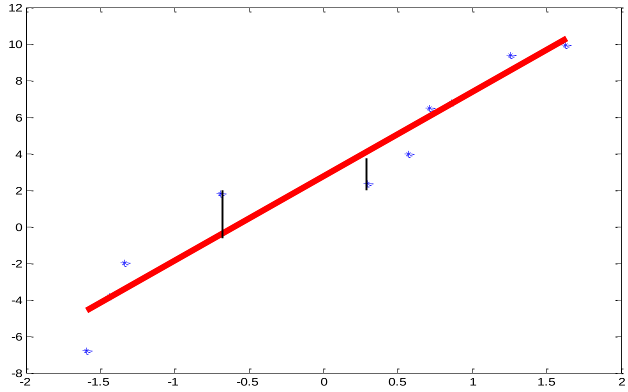
Training vs Generalization error

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



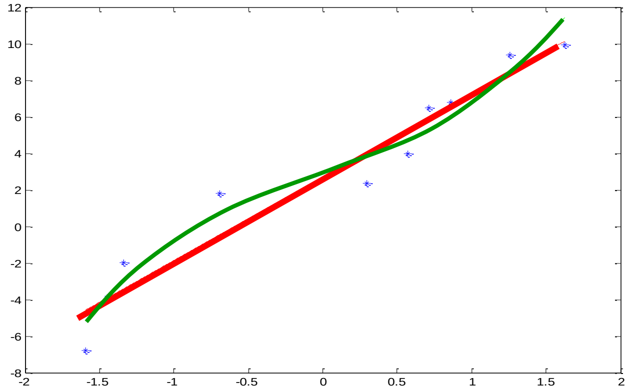
Training vs Generalization error

- Fitting a linear function with the square error
- Error is nonzero



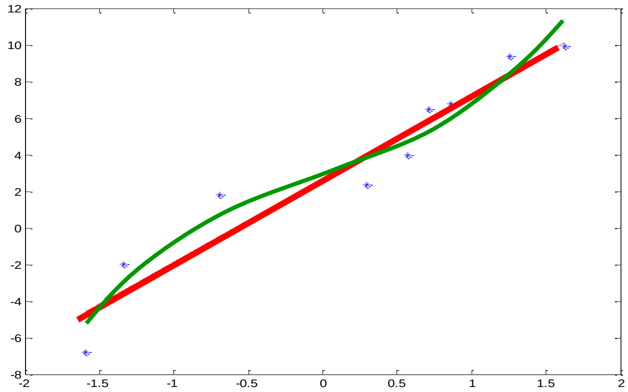
Training vs Generalization error

- Linear vs. cubic polynomial
-



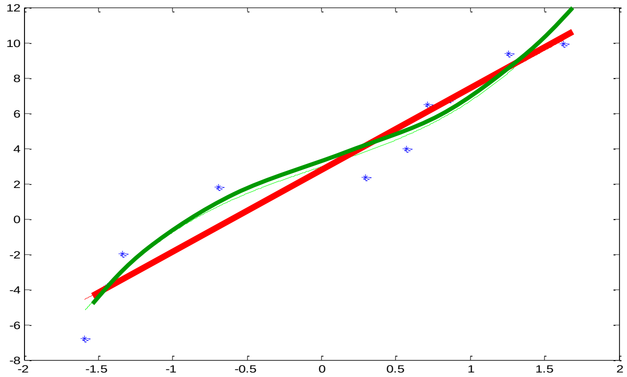
Training vs Generalization error

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



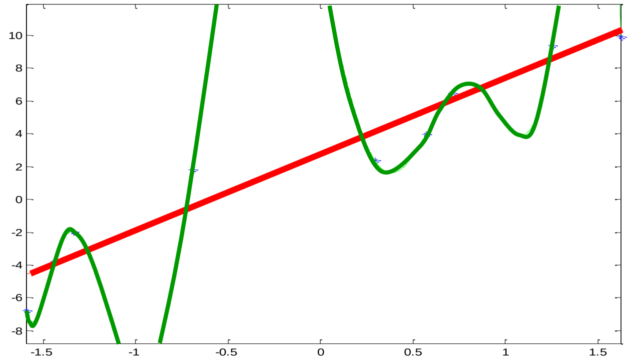
Training vs Generalization error

- Is it always good to minimize the error of the observed data?
- Remember: our goal is to optimize future errors



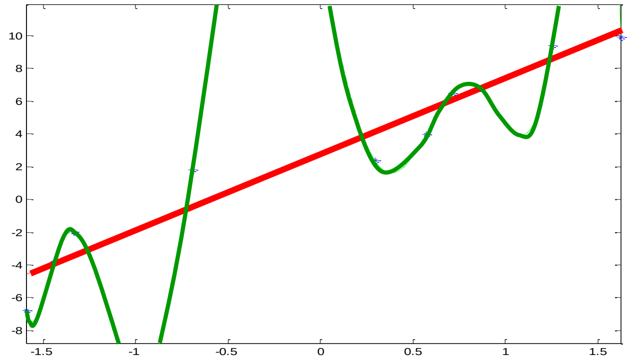
Training vs Generalization error

- 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

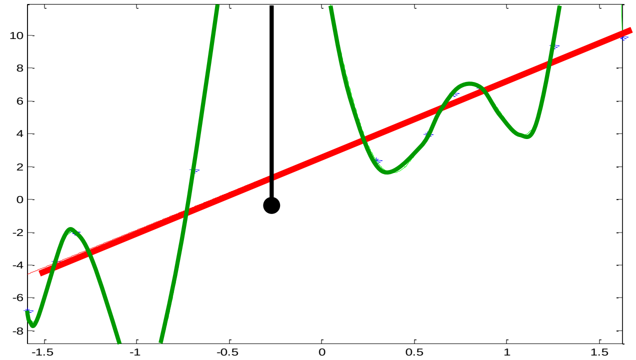
- For 10 data points, degree 9 polynomial gives a perfect fit (Lagrange interpolation). Error is zero.
- Is it always good to minimize the training error? NO !!
- **More important:** How do we perform on the unseen data?



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 test the generalization error?

How to evaluate the learner's performance?

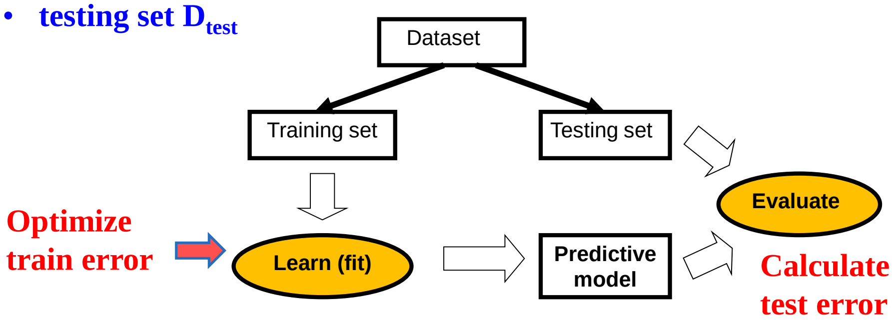
- **Generalization error** is the true error for the population of examples we would like to optimize
- **Sample mean only approximates it**
- **Two ways to assess the generalization error is:**
  - **Theoretical: Law of Large numbers**
    - statistical bounds on the difference between true generalization and sample mean errors
  - **Practical:** Use a separate data set with m data samples to test the model
    - **(Average) test error**

$$Error(D_{test}, f) = \frac{1}{m} \sum_{j=1,..m} (y_j - f(x_j))^2$$

Evaluation of the generalization performance

Split available data D into two disjoint sets:

- **training set D_{train}**
- **testing set D_{test}**



Also called: Simple holdout method

- Typically 2/3 training and 1/3 testing

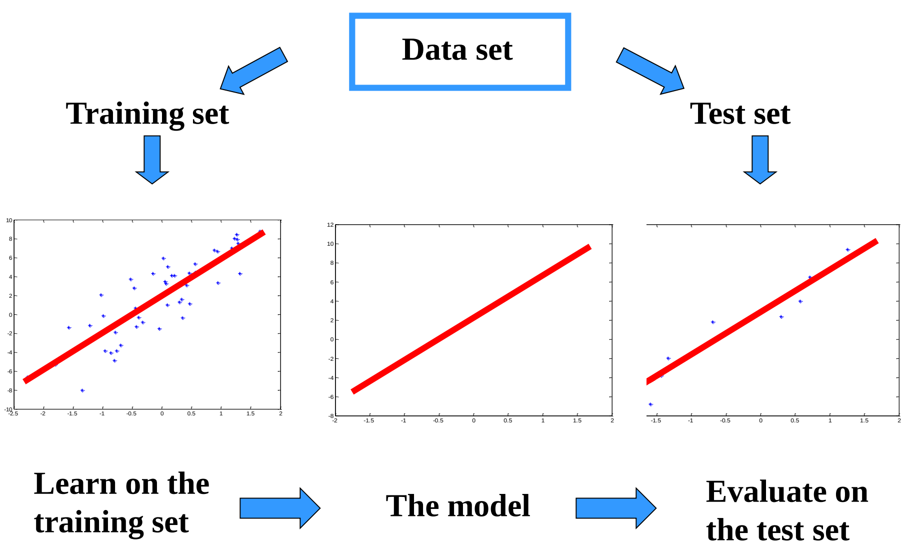
Assessment of model performance

Assessment of the generalization performance of the model:

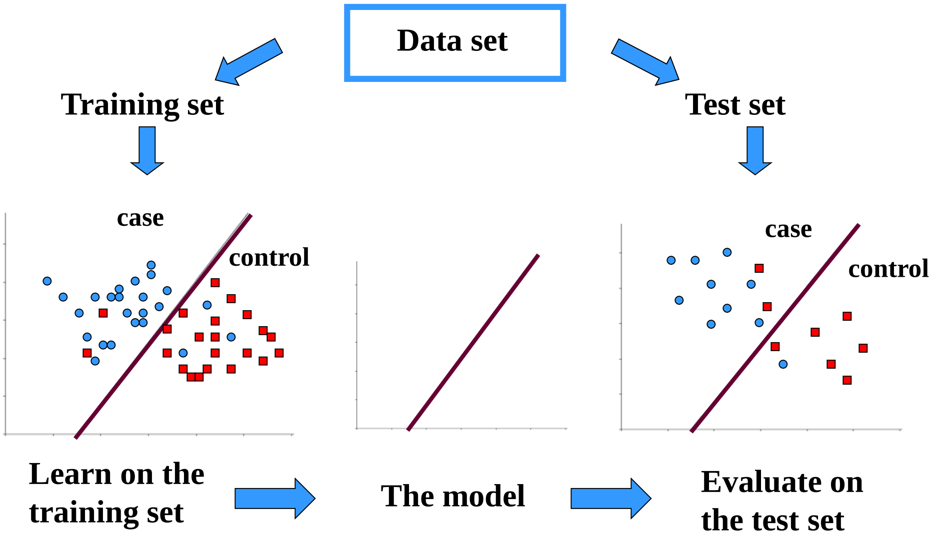
Basic rule:

- **Never ever touch the test data during the learning/model building process**
- **Test data should be used for the final evaluation only**

Testing of models: regression



Testing of models: classification



Evaluation measures

Easiest way to evaluate the model:

- **Error function used in the optimization is adopted also in the evaluation**
- **Advantage: may help us to see model overfitting. Simply compare the error on the training and testing data.**

Evaluation of the models often considers:

- **Other aspects or statistics of the model and its performance**
- Moreover the Error function used for the optimization may be a convenient approximation of the quality measure we would really like to optimize

Evaluation measures: classification

Binary 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:

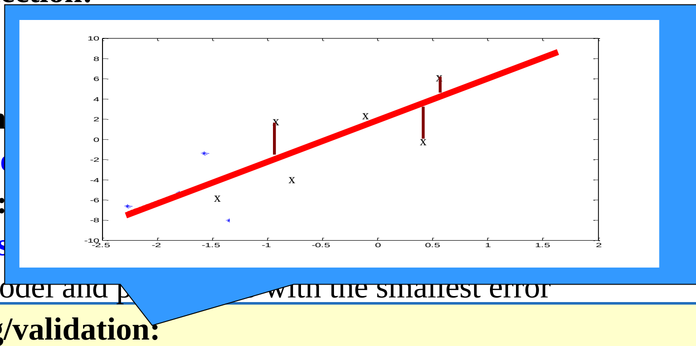
$$SP = \frac{TN}{TN + FP}$$

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

- Data
- Model selection
- Choice of Error function
- Learning/optimization
- Evaluation
- Application

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==
- Data cleaning/preprocessing
- Feature selection/dimensionality reduction
- Model selection
- Choice of Error function
- Learning/optimization
- Evaluation
- Application

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

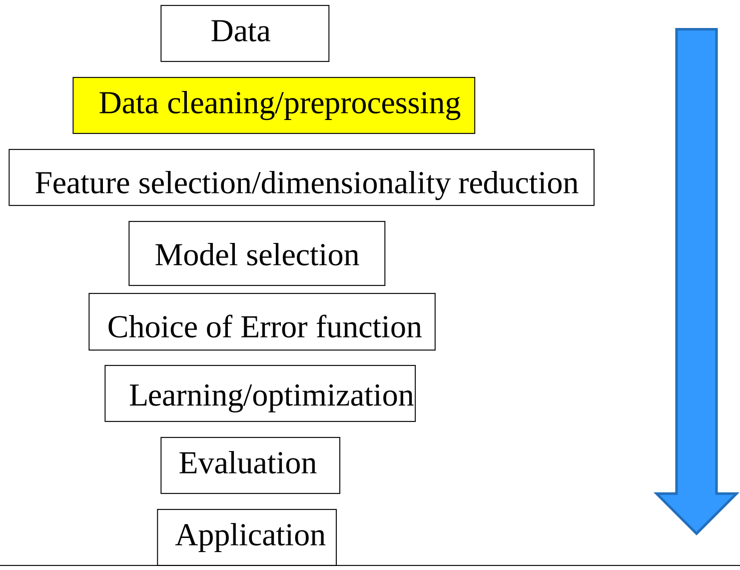
Data extraction:

- 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

Example:

- assume the following encoding of values High, Normal, Low

High → 2
Normal → 1
Low → 0

- **2 >1 implies High > Normal: Is it OK?**
- **1 > 0 implies Normal > Low: Is it OK?**
- **2 > 0 implies High > Low: Is it OK?**

?

Data preprocessing

Renaming (relabeling) categorical values to numbers

- dangerous in conjunction with some learning methods
- numbers will impose an order that is not warranted

Example:

- assume the following encoding of values High, Normal, Low

High → 2
Normal → 1
Low → 0

- **2 >1 implies High > Normal: Is it OK?**
- **1 > 0 implies Normal > Low: Is it OK?**
- **2 > 0 implies High > Low: Is it OK?**

√

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

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

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

Data preprocessing

Renaming (relabeling) categorical values to numbers

Problem: How to safely represent the different categories as numbers when no order exists?

Solution:

- **Use indicator vector (or one-hot) representation.**
- **Example: Red, Blue, Green colors**
  - **3 categories** → use a vector of size 3 with binary values
  - Encoding:
    - **Red:** (1,0,0);
    - **Blue:** (0,1,0);
    - **Green:** (0,0,1)