CS 1675 Introduction to ML

Lecture 2

Introduction to Machine Learning

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

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

Administration

Instructor:

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

TA:

Amin Sobhani
ams543@pitt.edu
6804 Sennott Square

Homework assignment

Homework assignment 1 is out

- Two parts:
  - **Programs**
  - **Report**
- Programs and the report should be submitted via Courseweb
- Deadline 4:00pm (prior to the lecture)

Rules:

- Strict deadline
- No collaboration on the programming and the report part

Machine Learning

- The field of **machine learning** studies the design of computer programs (agents) capable of learning from past experience or adapting to changes in the environment

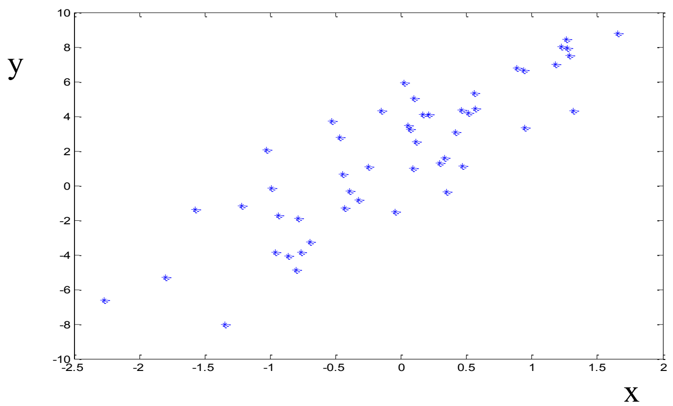
- The need for building agents capable of learning is everywhere
  - text, web page, image classification
  - web search
  - speech recognition
  - Image/video annotation and retrieval
  - adaptive interfaces
  - commercial software

Types of learning problems

- **Supervised learning**
  - Takes data that consists of pairs (**x,y**)
  - Learns mapping ***f:*** **x** (input) → **y** (output, response)
- **Unsupervised learning**
  - Takes data that consist of vectors **x**
    - Learns relations **x** among vector components
    - Groups/clusters data into the groups
- **Reinforcement learning**
  - Learns mapping ***f:*** **x** (input) → **y** (desired output)
  - From (**x,y,**r) triplets where **x** is an input, **y** is a response chosen by the user/system, and r is a reinforcement signal
  - **Online:** see x, choose y and observe r
- **Other types of learning:** **Active learning, Transfer learning, Deep learning**

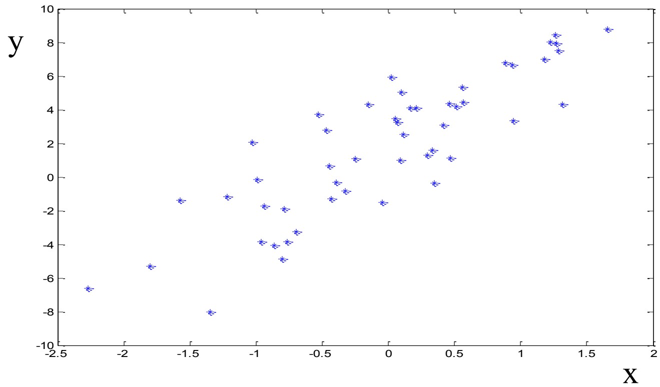
Learning: first look

- Assume we see examples of pairs (**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?



Supervised learning: regression

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

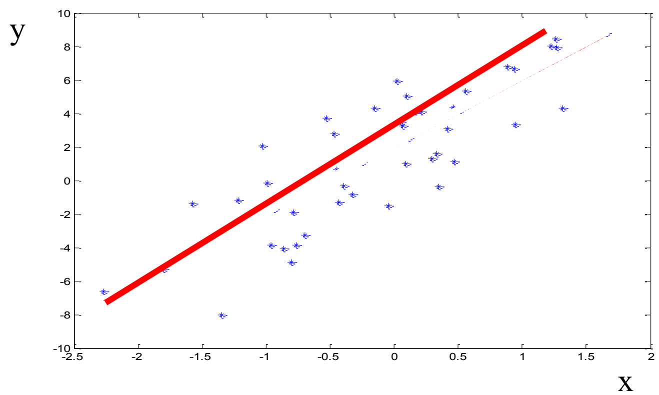


Supervised learning: regression

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

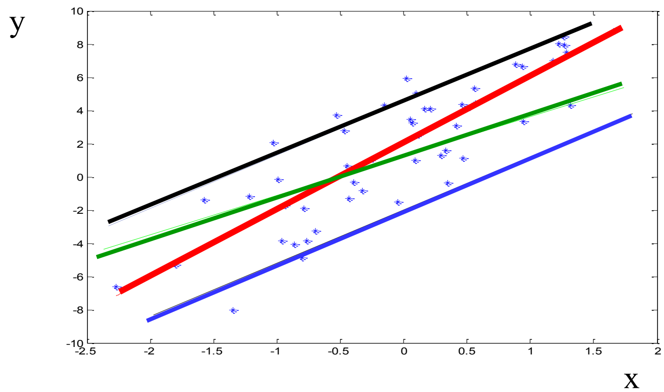
$$f(x) = ax + b$$

- An example of one such function:



Supervised learning: regression

- 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*



Fitting the data to the model

- We want the **best set** of model parameters

Objective: Find parameters that:

- reduce the misfit between the model **M** and observed data ***D***
- Or, (in other words) explain the data the best

Objective function:

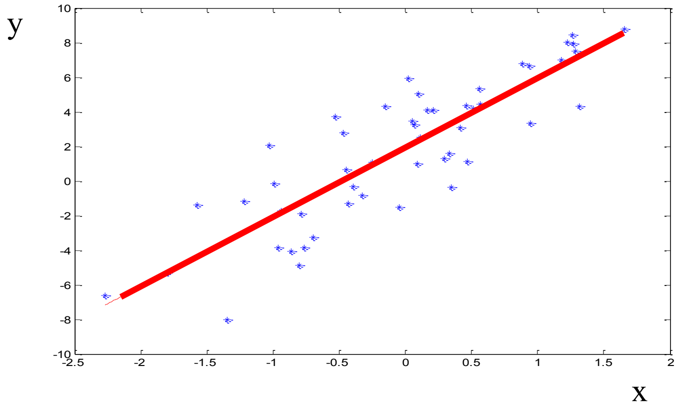
- **Error function: Measures the misfit between *D* and M**
- **Examples of error functions:**
  - Average Square Error $\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$
  - Average misclassification error $\frac{1}{n}\sum_{i=1}^{n} 1_{y_i \neq f(x_i)}$

    Average # of misclassified cases

Fitting the data to the model

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

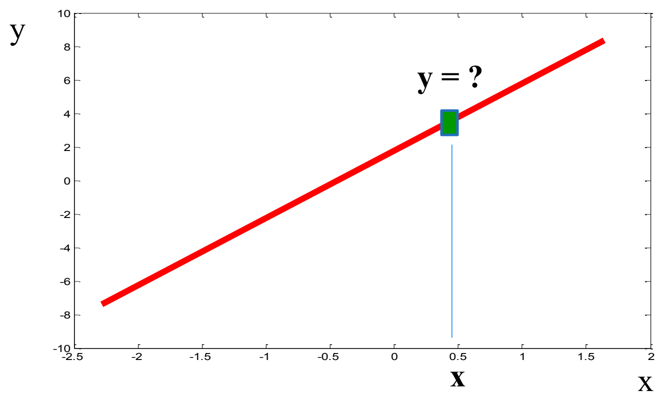
$$\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$$



Supervised learning: Regression

- **Application:** A new example **x** with unknown value y is checked against the model, and y is calculated

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

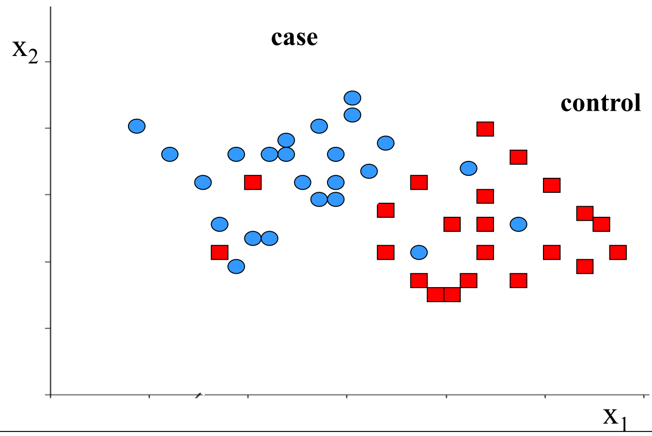


Supervised learning: Classification

- **Data** D: pairs (**x** , *y*) where y is a class label:

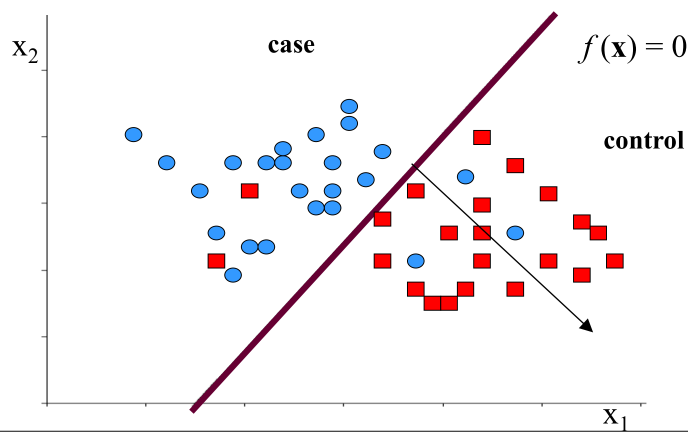
  y examples: patient will be readmitted or no,

  has disease (case) or no (control)



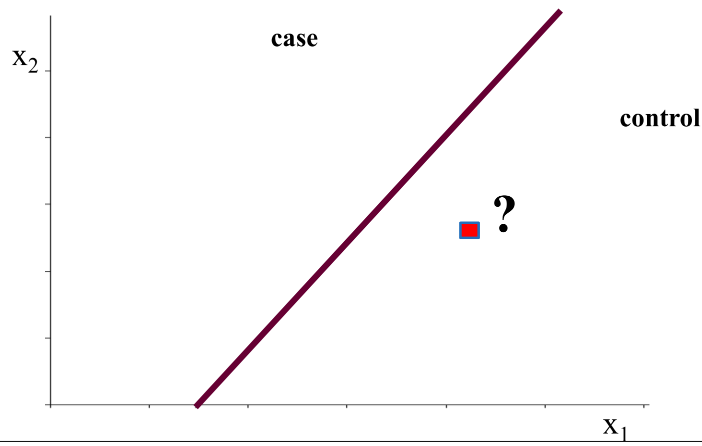
Supervised learning: Classification

- Find a model *f*: X → R, say $f(x) = ax_1 + bx_2 + c$ that defines a decision boundary $f(\mathbf{x}) = 0$ that separates well the two classes
  - **Note that some examples are not correctly classified**



Supervised learning: Classification

- A new example x with unknown class label is checked against the model, the class label is assigned



Learning: summary

Three basic steps:

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

  E.g. $f(x) = ax + b$

- **Select the error function** to be optimized

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

- **Find the set of parameters optimizing the error function**
  - The model and parameters with the smallest error represent the best fit of the model to the data

But there are problems one must be careful about …

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):

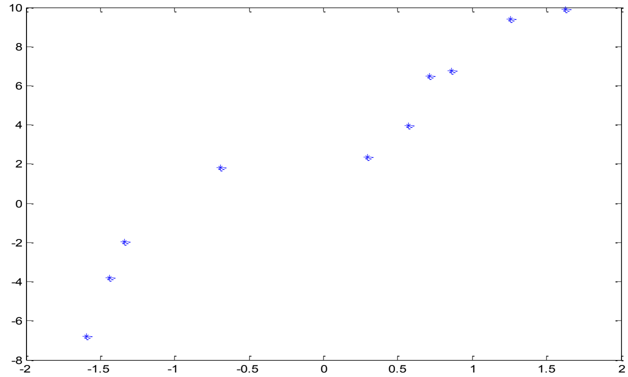
$$E_{(x,y)}[(y - f(x))^2]$$ Mean squared error

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

Does a good training error imply a good generalization error ?

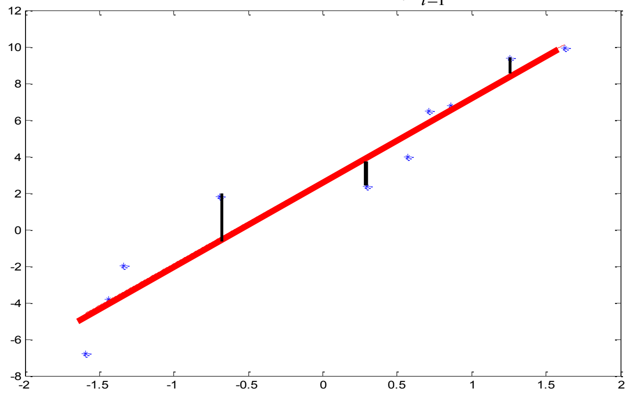
Overfitting

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

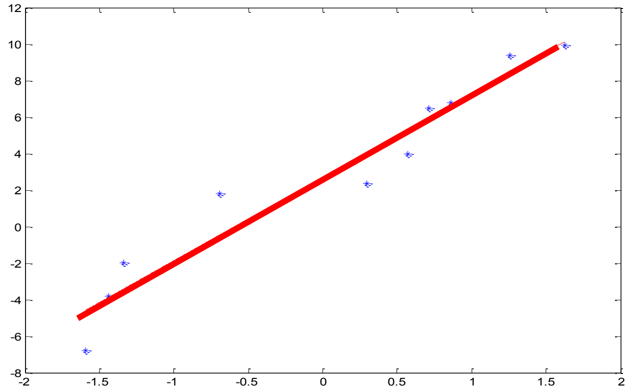


Overfitting

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

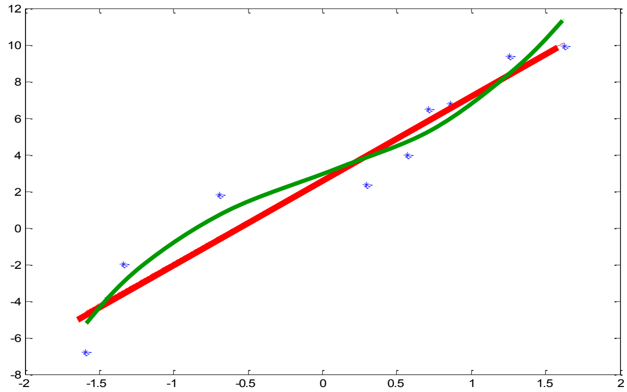
we consider also: $y = f(x) = a_3x^3 + a_2x^2 + a_1x + b$

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



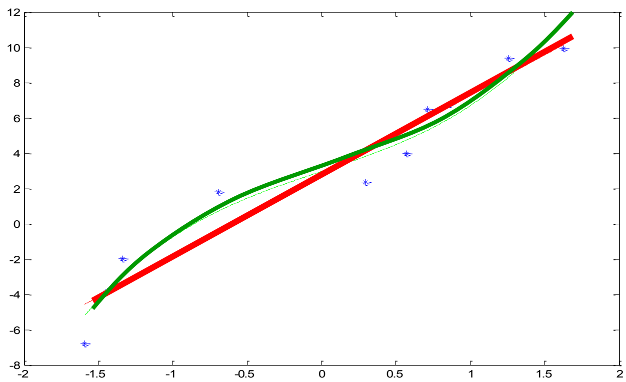
Overfitting

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



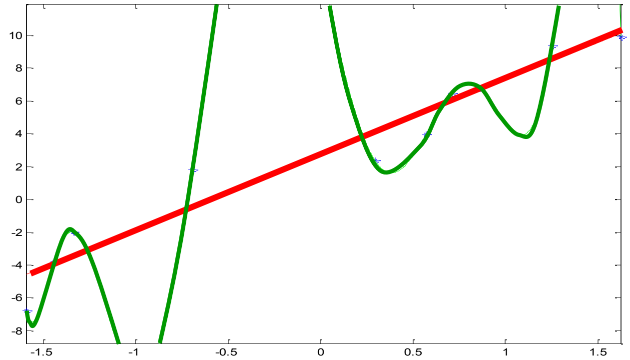
Overfitting

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



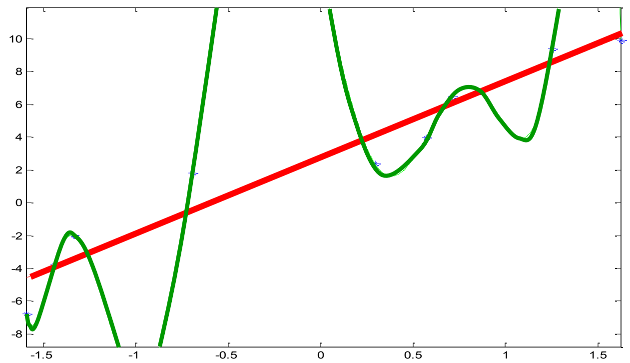
Overfitting

- 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

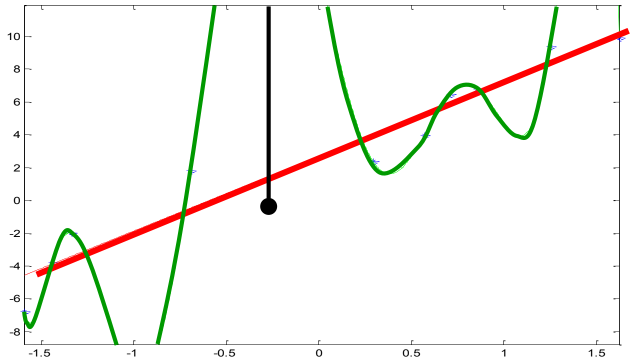
- For 10 data points, degree 9 polynomial gives a perfect fit (Lagrange interpolation). Error is zero.
- Is it always good to minimize the training error? NO !!
- **More important:** How do we perform on the unseen data?



Overfitting

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

- 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 properly reflect the generalization error

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

- So how to assess the generalization error?

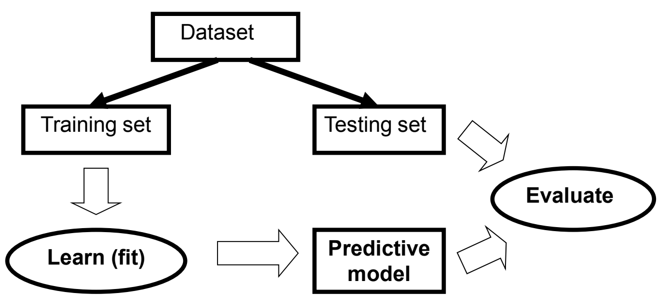
How to evaluate the learner's performance?

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

CS 2750 Machine Learning

Testing of learning models

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



  - Typically 2/3 training and 1/3 testing

CS 2750 Machine Learning

Testing of models