CS 1571 Introduction to AI
Lecture 23

Machine learning

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

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
  - predictions in medicine,
  - text and web page classification,
  - speech recognition,
  - image/text retrieval,
  - commercial software

Learning

Learning process:

Learner (a computer program) processes data ***D*** representing past experiences and tries to either to develop an appropriate response to future data, or describe in some meaningful way the data seen

Example:

Learner sees a set of patient cases (patient records) with corresponding diagnoses. It can either try:

- to predict the presence of a disease for future patients
- describe the dependencies between diseases, symptoms

Types of learning

- **Supervised learning**
  - Learning mapping between inputs x and desired outputs y
  - Teacher gives me y's for the learning purposes
- **Unsupervised learning**
  - Learning relations between data components
  - No specific outputs given by a teacher
- **Reinforcement learning**
  - Learning mapping between inputs x and desired outputs y
  - Critic does not give me y's but instead a signal (reinforcement) of how good my answer was
- **Other types of learning:**
  - **explanation-based learning, etc.**

Supervised learning

Data: $D = \{d_1, d_2, .., d_n\}$ **a set of *n* examples**

$d_i = < \mathbf{x}_i, y_i >$

$\mathbf{x}_i$ is input vector, and *y* is desired output (given by a teacher)

Objective: learn the mapping $f : X \rightarrow Y$

s.t. $y_i \approx f(x_i)$ for all $i = 1, .., n$

Two types of problems:

- **Regression:** X discrete or continuous → Y is **continuous**
- **Classification:** X discrete or continuous → Y is **discrete**

Supervised learning examples

- **Regression:** Y is **continuous**

Debt/equity
Earnings
Future product orders

→ company stock price

- **Classification:** Y is **discrete**

→ Label "3"

Handwritten digit (array of 0,1s)

Unsupervised learning

- **Data:** $D = \{d_1, d_2, .., d_n\}$
  $d_i = \mathbf{x}_i$ vector of values
  No target value (output) y

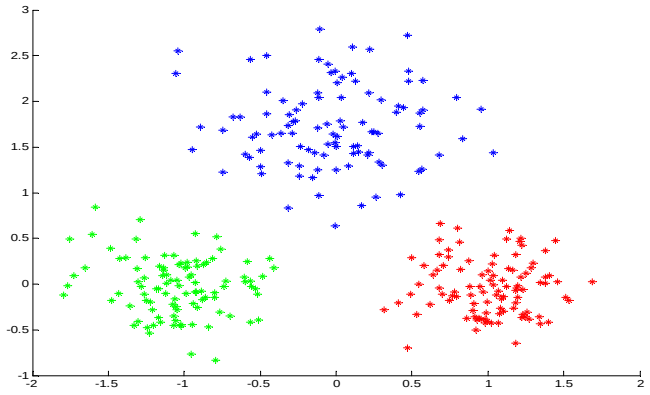
- **Objective:**
  - learn relations between samples, components of samples

Types of problems:

- **Clustering**
  Group together "similar" examples, e.g. patient cases
- **Density estimation**
  - Model probabilistically the population of samples

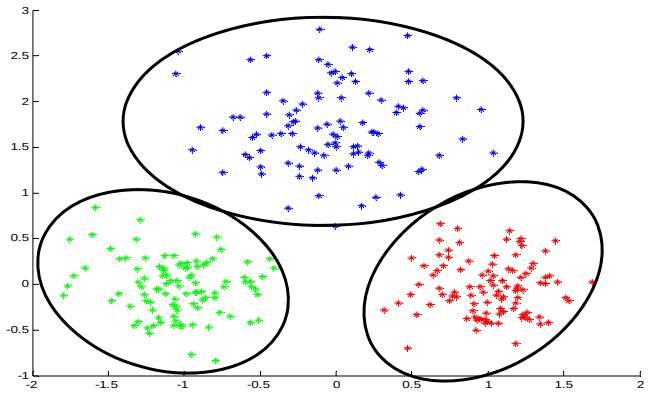
Unsupervised learning example

- **Clustering.** Group together similar examples $d_i = \mathbf{x}_i$



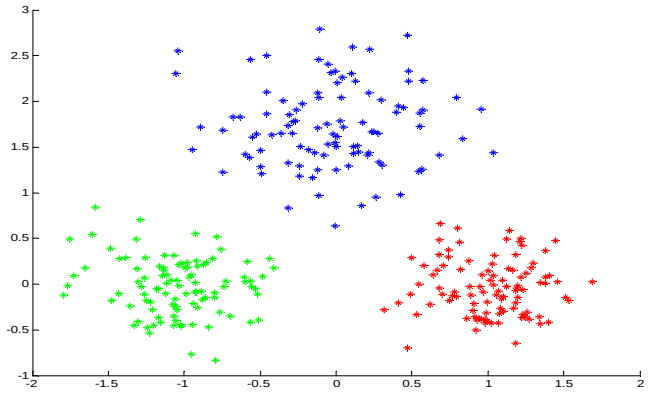
Unsupervised learning example

- **Clustering.** Group together similar examples $d_i = \mathbf{x}_i$



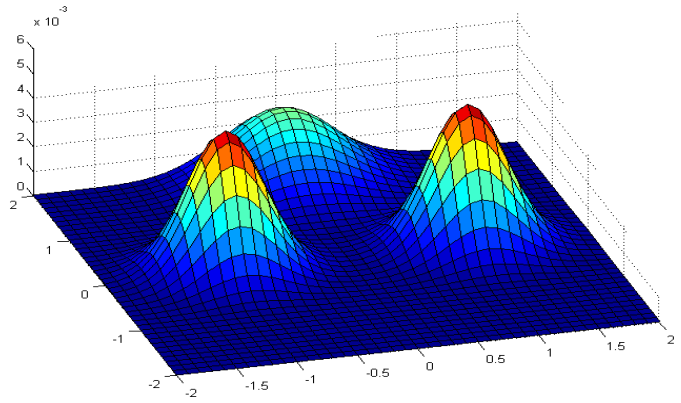
Unsupervised learning example

- **Density estimation.** We want to build the probability model P(**x**) of a population from which we draw examples $d_i = \mathbf{x}_i$



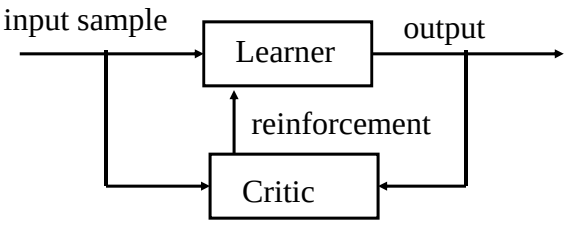
Unsupervised learning. Density estimation

- A probability density of a point in the two dimensional space
  - Model used here: **Mixture of Gaussians**



Reinforcement learning

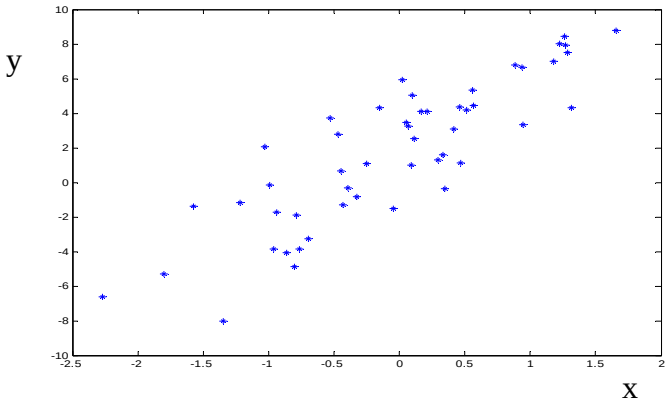
- We want to learn: $f : X \rightarrow Y$
- We see samples of **x** but not *y*
- Instead of *y* we get a feedback (reinforcement) from a **critic** about how good our output was



- The goal is to select output that leads to the best reinforcement

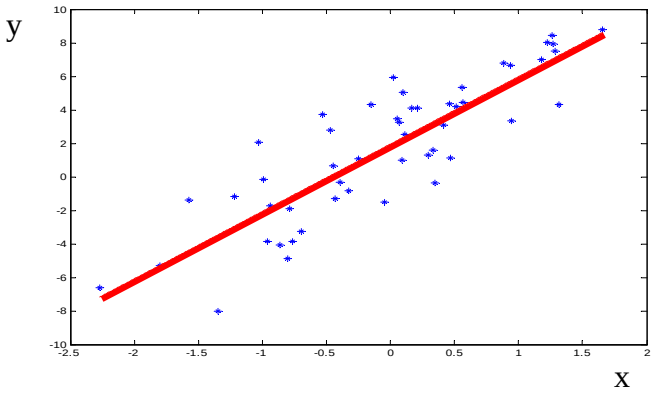
Learning

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



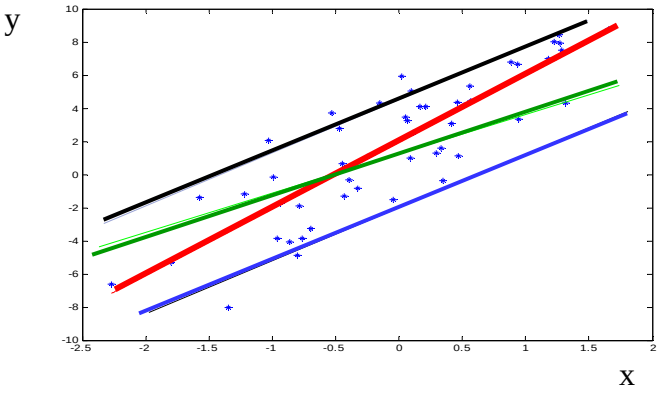
Learning bias

- Problem is easier when we make an assumption about the model, say, $f(x) = ax + b$
- Restriction to a linear model narrows down the possibilities



Learning bias

- Choosing a parametric model $f(x) = ax + b$
- Many possible functions: One for every pair of parameters a, b



Fitting the data to the model

- We are interested in finding the **best set** of model parameters

Objective: Find the set of parameters that:

- improve the fit between what model suggests and what data say
- Or, (in other words) that explain the data the best

Error function:

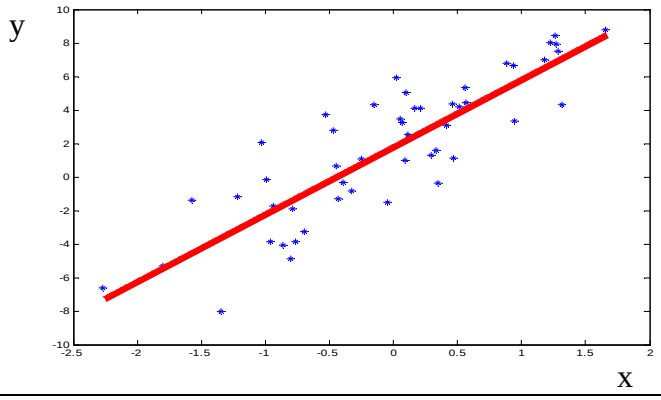
Measure of misfit between the data and the model

- Examples of error functions:
  - Mean square error $\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$
  - Misclassification error

Average # of misclassified cases $y_i \neq f(x_i)$

Fitting the data to the model

- **Linear regression**
  - Least squares fit with the linear model
  - minimizes $\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$



Typical learning

Three basic steps:

- **Select a model** or a set of models (with parameters)
  E.g. $y = ax + b + \varepsilon \quad \varepsilon = N(0, \sigma)$
- **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

Problem

- We fit the model based on past experience (past examples seen)
- But ultimately we are interested in learning the mapping that performs well on the whole population of examples

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

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

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

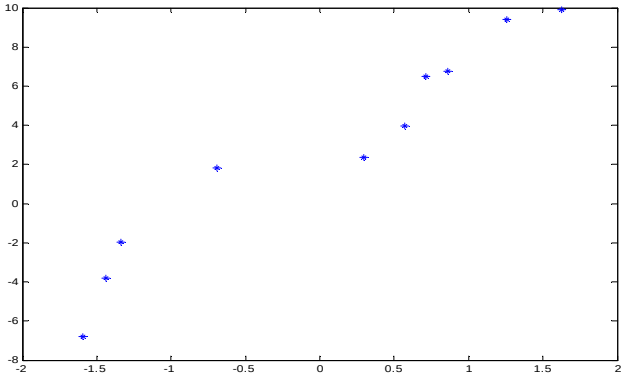
$E_{(x,y)}(y - f(x))^2$ Expected squared error

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

Does a good training error imply a good generalization error ?

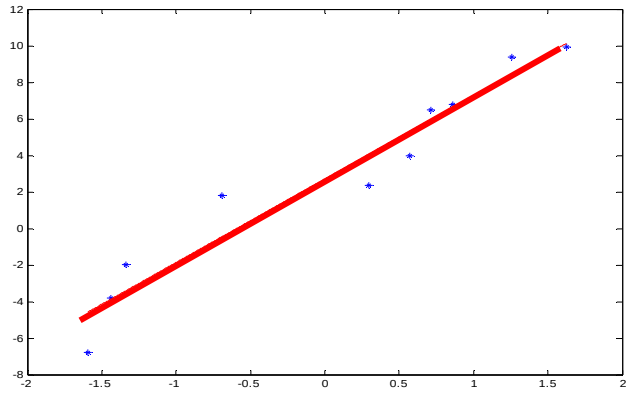
Overfitting

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



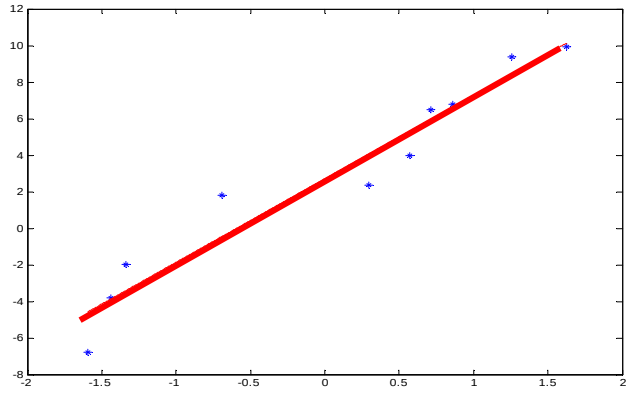
Overfitting

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



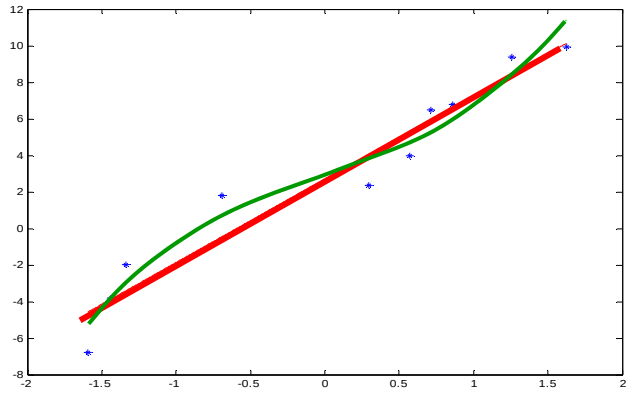
Overfitting

- Fitting a linear function with mean-squares error
- Error is nonzero



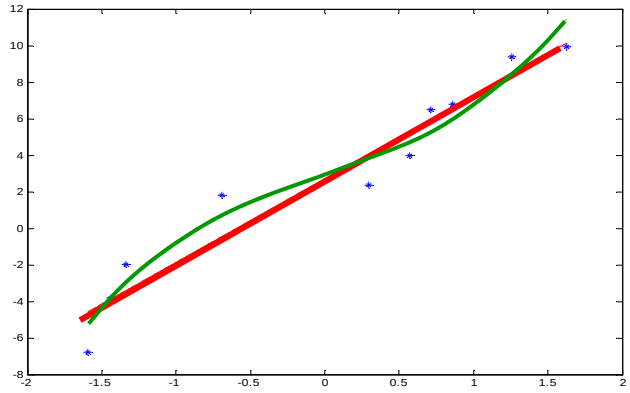
Overfitting

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



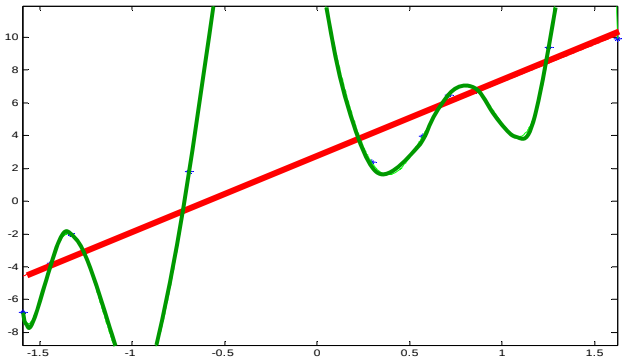
Overfitting

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



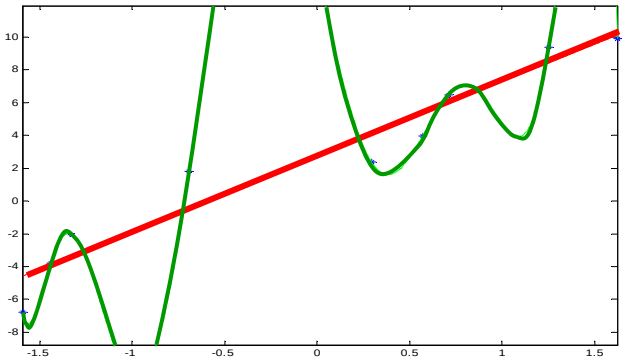
Overfitting

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



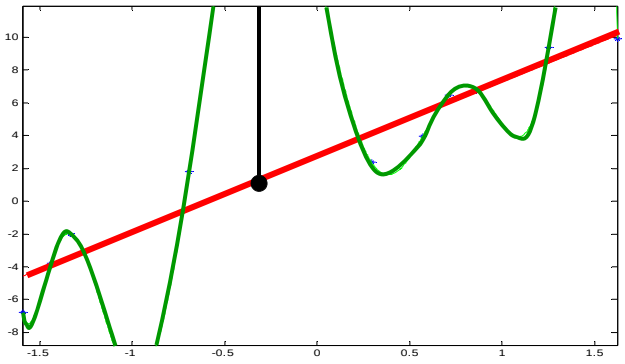
Overfitting

- 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 more degrees of freedom (more parameters)
  - Small data size (as compared to the complexity of 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**
- **Optimizing (mean) training error can lead to 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 assess the learner's performance?

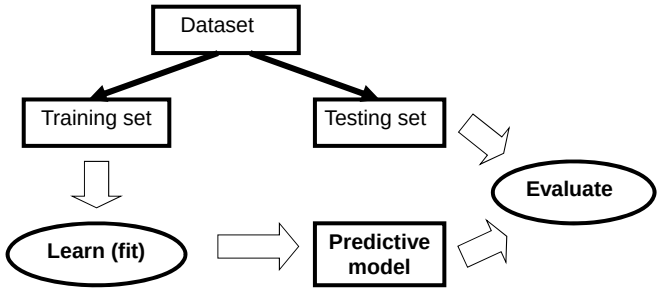
- **Generalization error** is the true error for the population of examples we would like to optimize

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

- **Sample mean only approximates it**
- How to measure the generalization error?
- **Two ways:**
  - **Theoretical: Law of Large numbers**
    - statistical bounds on the difference between the true and sample mean errors
  - **Practical:** Use a separate data set with *m* data samples to test
    - **(Mean) test error** $\frac{1}{m}\sum_{j=1,..m}(y_j - f(x_j))^2$

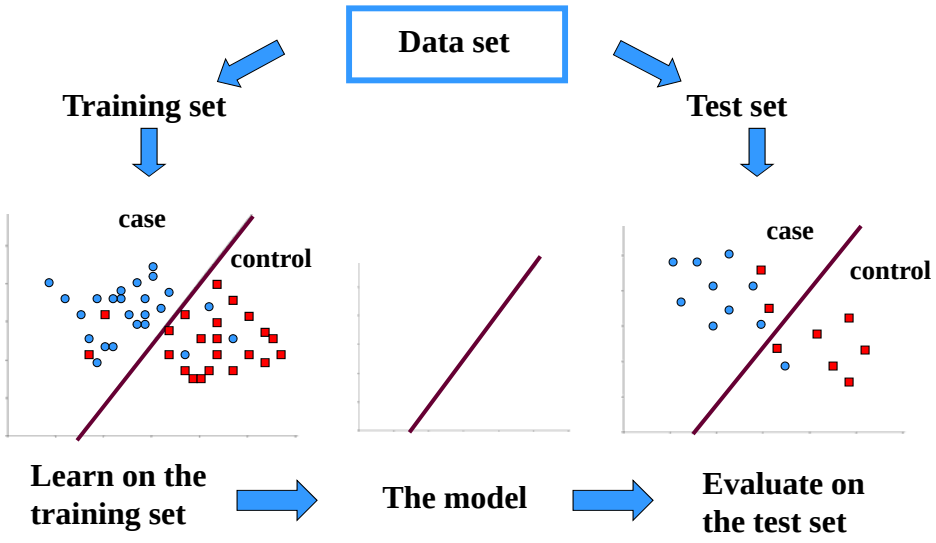
Testing of learning models

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



  - Typically 2/3 training and 1/3 testing

Testing of models



Basic experimental setup to test the learner's performance

1. Take a dataset D and divide it into:

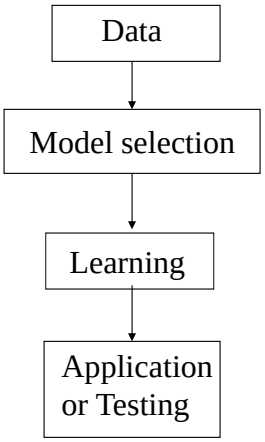
- **Training data set**
- **Testing data set**

2. Use the training set and your favorite ML algorithm to train the learner

3. Test (evaluate) the learner on the testing data set

- The results on the testing set can be used to compare different learners powered with different models and learning algorithms

Design of a learning system (first view)



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:

- **Apply the learned model to new data**
- E.g. predict ys for new inputs **x** using learned $f(\mathbf{x})$
- Evaluate on the test data

A learning system: basic cycle

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**

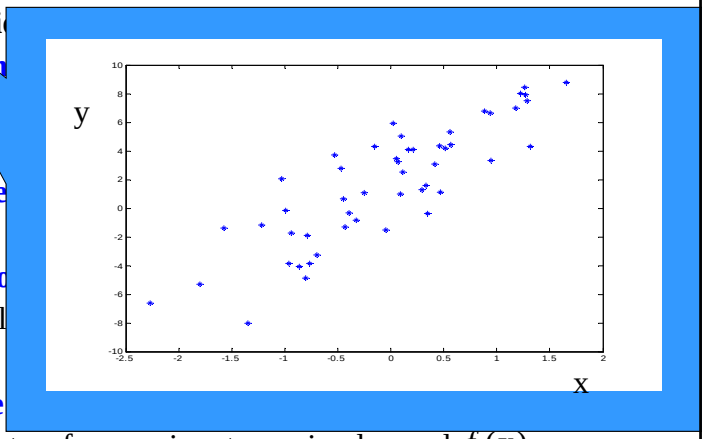
– The model

5. Testing:

– **Apply the**

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

– Evaluate on the test data



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

– **Squar**

4. Learning:

• **Find the s**

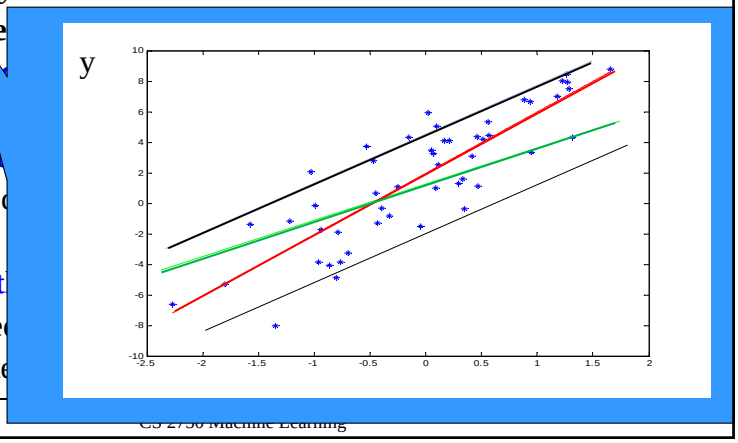
– The mod

5. Testing:

– **Apply t**

– E.g. pre

– Evaluate



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 se**

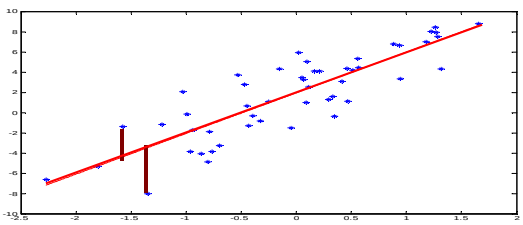
– The mo

5. Testing:

– **Apply t**

– E.g. pre

– Evaluate



A learning system: basic cycle

1. Data: $D =$

2. Model sele

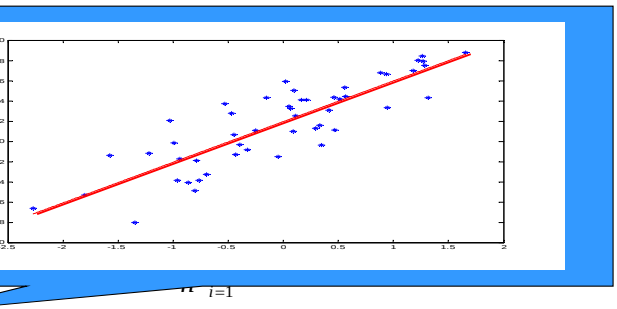
– **Select a**

E.g. y

3. Choose the

– **Squared**

4. Learning:



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

– The model and parameters with the smallest error

5. Testing:

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

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

– Evaluate on the test data