

CS 1675 Introduction to Machine Learning

Lecture 1

Machine Learning

Milos Hauskrecht

milos@pitt.edu

5329 Sennott Square, x4-8845

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

Administration

Instructor:

Prof. Milos Hauskrecht

milos@pitt.edu

5329 Sennott Square, x4-8845

TA:

Jeongmin Lee

jlee@cs.pitt.edu

5324 Sennott Square, x4-8455

Office hours: TBA

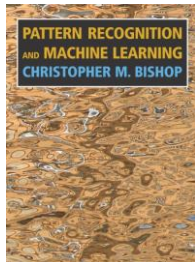
Who am I?

- **Milos Hauskrecht –Professor of Computer Science**
 - **Secondary affiliations:**
 - Intelligent Systems Program (ISP),
 - Department of Biomedical Informatics (DBMI)
 - **Research work:**
 - Machine learning, Data mining, Outlier detection, Probabilistic modeling, Time-series models and analysis
- Applications to healthcare:**
- EHR data analysis, Patient monitoring and alerting, Patient safety
-

Administration

Study material

- Handouts, your notes and course readings
- **Primary textbook:**



Chris. Bishop. Pattern Recognition and Machine Learning. Springer, 2006.

- **Additional text/readings:**

Hal Daume III. A Course in Machine Learning. <http://ciml.info>
-

Administration

Study material

- **Other ML books:**

- K. Murphy. Machine Learning: A probabilistic perspective, MIT Press, 2012.
 - J. Han, M. Kamber. Data Mining. Morgan Kauffman, 2011.
 - Friedman, Hastie, Tibshirani. Elements of statistical learning. Springer, 2nd edition, 2011.
 - Koller, Friedman. Probabilistic graphical models. MIT Press, 2009.
 - Duda, Hart, Stork. Pattern classification. 2nd edition. J Wiley and Sons, 2000.
 - T. Mitchell. Machine Learning. McGraw Hill, 1997.
-

Administration

- **Homework assignments (45%): weekly**

- **Programming tool:** Matlab (free license, CSSD machines and labs)
- **Matlab Tutorial:** recitations

- **Exams (45%):**

- **Midterm** – March 7, 2019
- **Final** – Final exam week

- **Lectures (10%):**

- **Attendance and Activity**
 - **Short quizzes**
-

Tentative topics

- **Introduction**
 - **Density estimation:**
 - basic parametric distributions
 - non-parametric density estimation methods
 - **Supervised Learning**
 - Linear models for regression and classification.
 - Multi-layer neural networks.
 - Support vector machines. Kernel methods.
 - Decision trees
 - Non-parametric classification models
 - **Probabilistic graphical models**
 - Bayesian belief networks
 - Learning of BBNs
-



Tentative topics (cont)

- **Clustering:**
 - K-means clustering
 - Hierarchical clustering
 - **Ensemble methods**
 - Mixture models
 - Bagging and boosting
 - **Dimensionality reduction**
 - Feature selection
 - Principal component analysis (PCA)
 - **Reinforcement learning**
-



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.

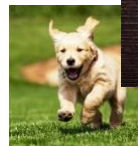
Examples:

- text, web-page classification
- web search
- speech recognition



Machine Learning

- **Examples (cont):**
 - Image/video classification, annotation and retrieval
 - adaptive interfaces
 - Route optimization
 - Game playing



Car
Race car



Learning process



Learner (a computer program) processes data D representing past experiences and tries to build a **model** that either:

- Generates appropriate response to future data, or
- Describes 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 occurrence of a disease for future patients
- describe the dependencies between diseases, symptoms

Types of learning problems

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

Supervised learning

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

$$d_i = \langle \mathbf{x}_i, y_i \rangle$$

\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 $i = 1, \dots, n$

Two types of problems:

- **Regression:** X discrete or continuous \rightarrow
 Y is **continuous**
- **Classification:** X discrete or continuous \rightarrow
 Y is **discrete**

Supervised learning examples

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

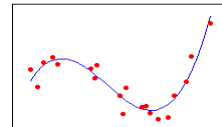
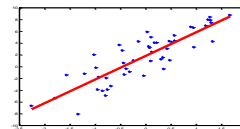
Debt/equity

Earnings

Future product orders



Stock price



Data:

Debt/equity	Earnings	Future prod orders	Stock price
20	115	20	123.45
18	120	31	140.56
....			

Supervised learning examples

- **Classification:** Y is discrete

```
#####
#
#
#####
#
#
#####
#
#
#####
```



Label “3”

Handwritten digit (array of 0,1s)



Data:

```
#####
#
#
#####
#
#
#####
#
#
#####
```



image



digit

3
7
5

....

Unsupervised learning

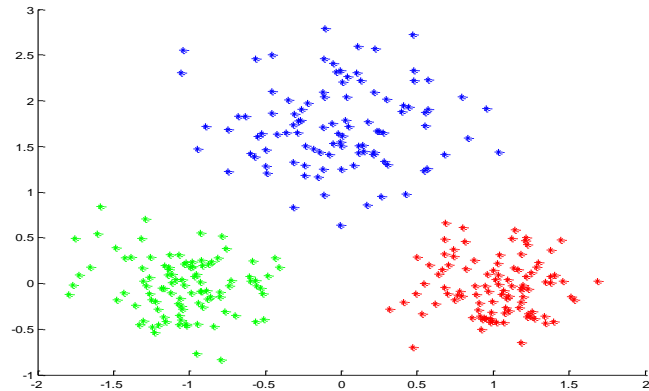
- **Data:** $D = \{d_1, d_2, \dots, d_n\}$
 $d_i = \mathbf{x}_i$ vector of values
 No target value (output) y
- **Objective:**
 - learn relations between samples, components of samples

Types of problems:

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

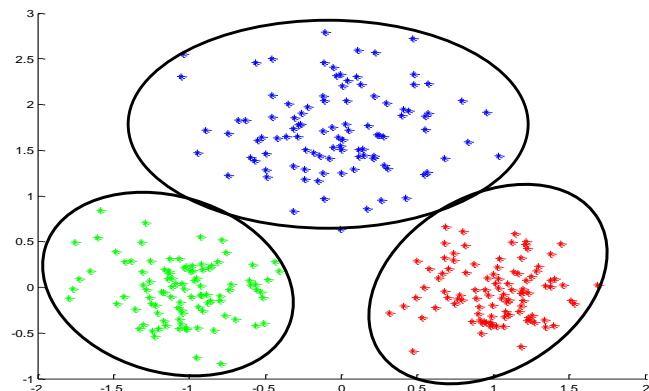
Unsupervised learning example

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



Unsupervised learning example

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



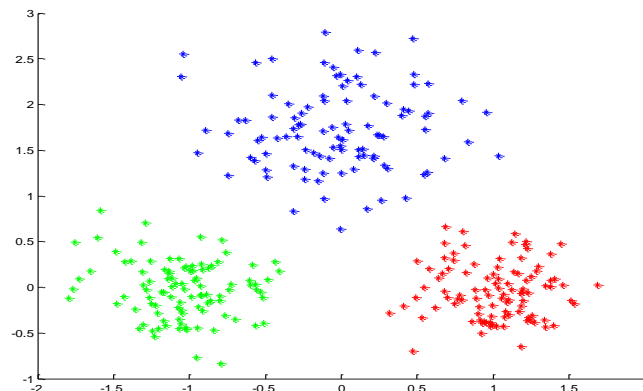
Unsupervised learning example

- **Clustering.** Group together similar examples



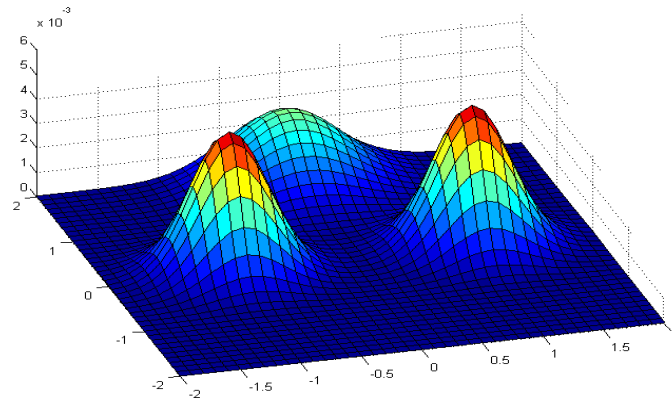
Unsupervised learning example

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



Unsupervised learning. Density estimation

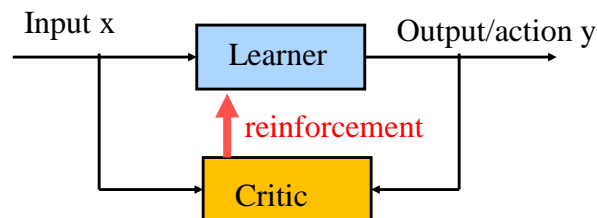
- A probability density of a point in the two dimensional space
 - Model used here: **Mixture of Gaussians**



Reinforcement learning

We want to learn: $f : X \rightarrow Y$

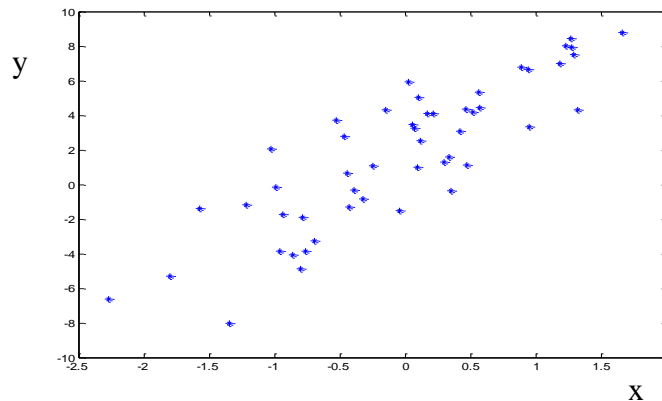
- We see examples of inputs \mathbf{x} but not y
- We select y for observed x from available choices
- We get a feedback (reinforcement) from a **critic** about how good our choice of y was



- The goal is to select outputs that lead to the best reinforcement

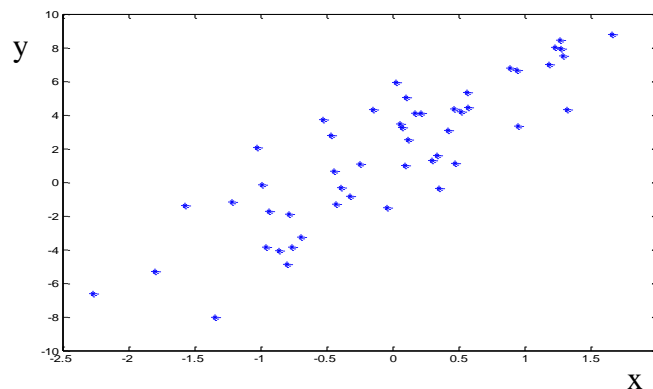
Learning: first look

- 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 \mathbf{x}
- We get the data D - what should we do?



Learning: first look

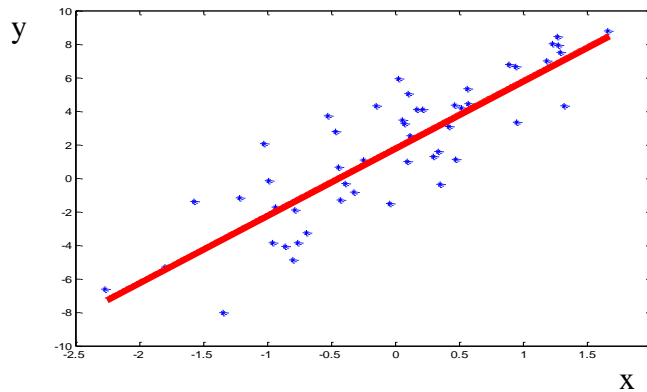
- **Problem:** many possible functions $f : X \rightarrow Y$ exists for representing the mapping between \mathbf{x} and y
- Which one to choose? Many examples still unseen!



Learning: first look

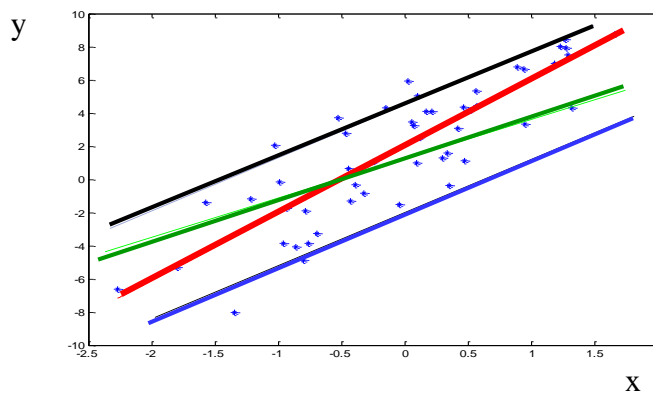
- **Solution:** make an assumption about the model, say,

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



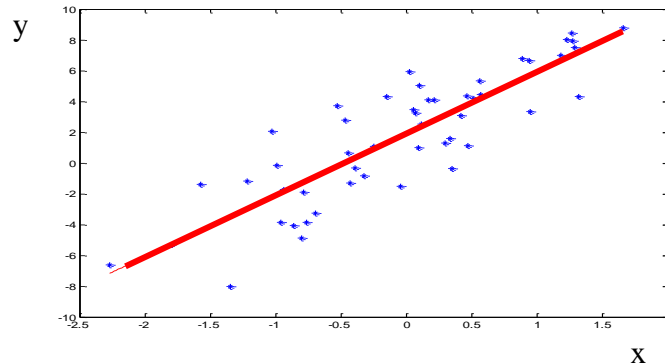
Learning: first look

- 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



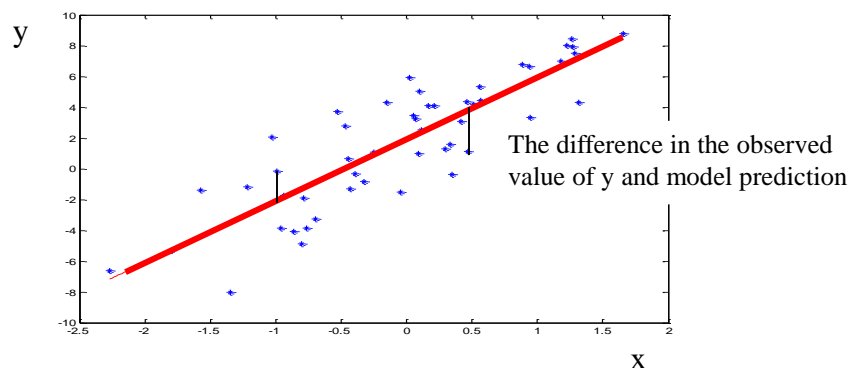
Learning: first look

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



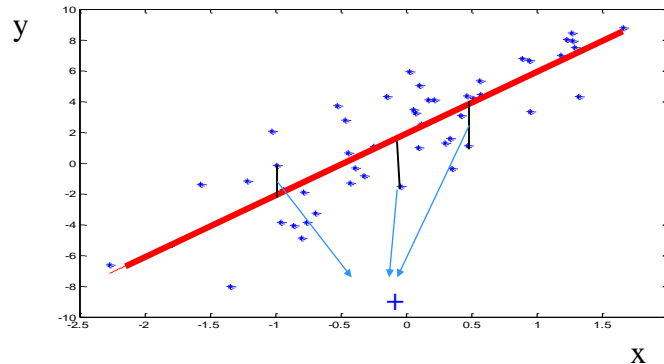
Learning: first look

- We want the **best set** of model parameters
 - reduce the misfit between the model \mathbf{M} and observed data \mathbf{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:**

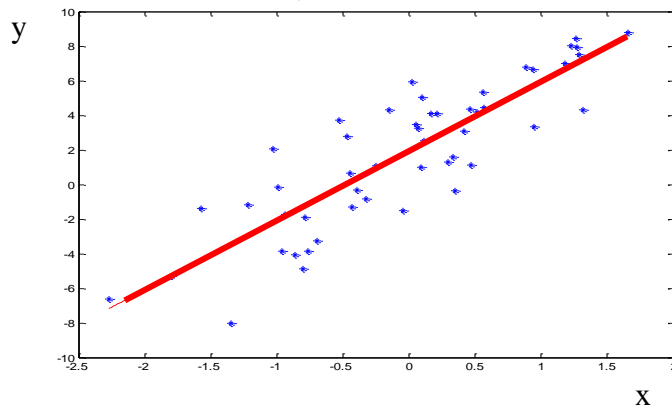
- Average Square Error
$$\frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$$

- 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

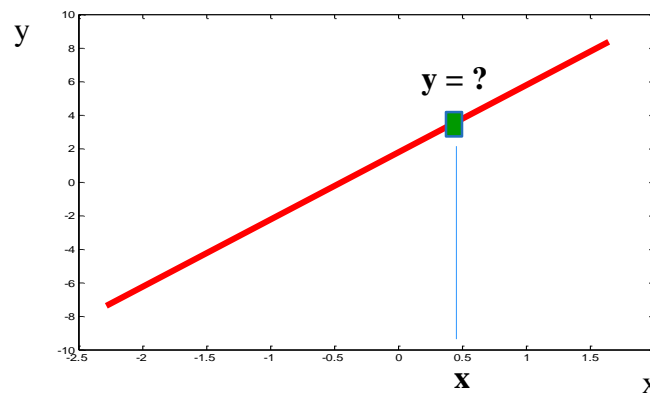
$$\frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$$



Learning: first look

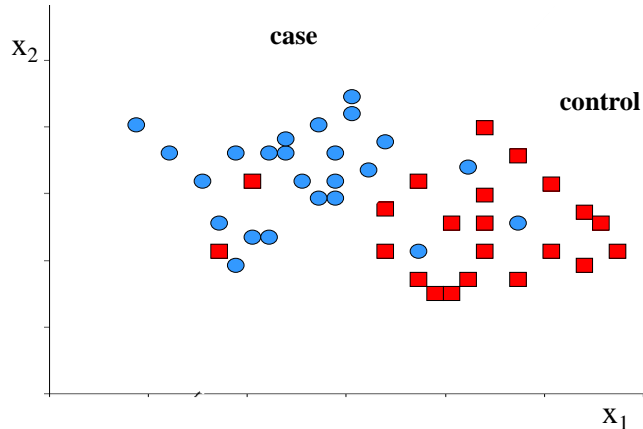
- **Application:** A new example \mathbf{x} with unknown value y is checked against the model, and y is calculated

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



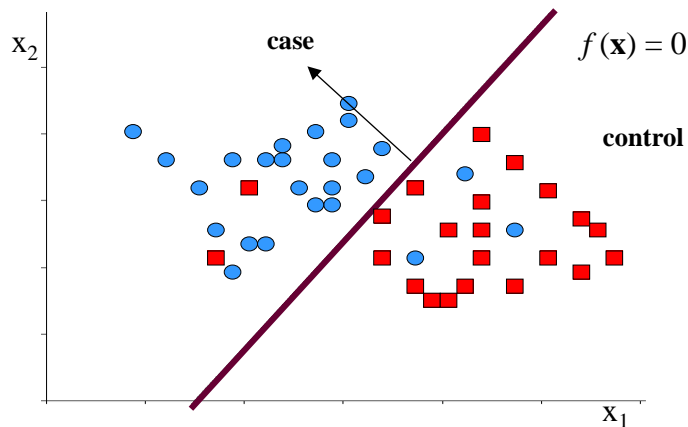
Supervised learning: Classification

- **Data D:** pairs (\mathbf{x}, y) where y is a class label:
 y examples: patient will be readmitted (case) or no (control),
 patient has disease (case) or no (control)



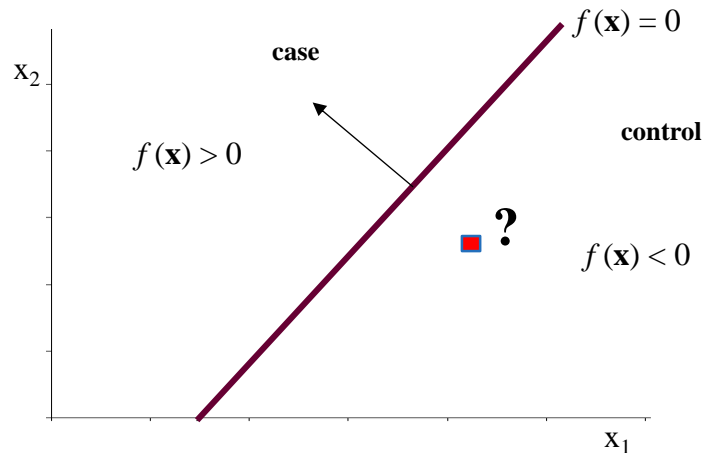
Supervised learning: Classification

- Find a model $f: X \rightarrow \mathbb{R}$, 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 may not be correctly classified**



Supervised learning: Classification

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



Learning: first look

1. **Data:** $D = \{d_1, d_2, \dots, d_n\}$
2. **Model selection:**
 - **Select a model** or a set of models (with parameters)
E.g. $y = ax + b$
3. **Choose the objective function (loss 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 y s for new inputs \mathbf{x} using learned $f(\mathbf{x})$

Learning: first look

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

2. Model selection:

- Select a model

E.g.

3. Choose the error

- Squared error

4. Learning:

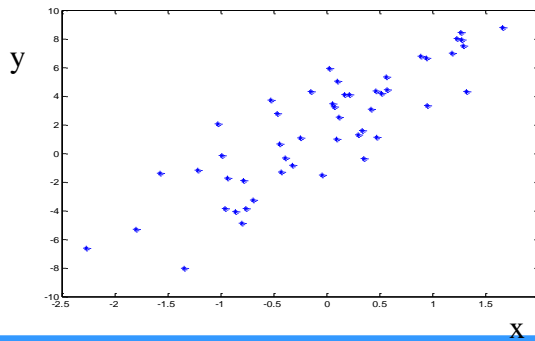
- Find the set of parameters

- The model

5. Application:

- Apply the model

- E.g. predict y s for new inputs x using learned $f(x)$



A learning system: basic cycle

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

2. Model selection:

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

E.g. $y = ax + b$

3. Choose the error

- Squared error

4. Learning:

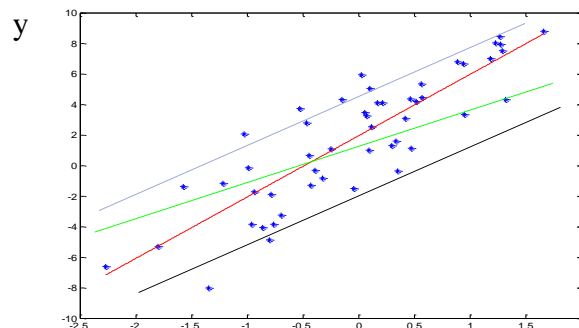
- Find the set of parameters

- The model

5. Application:

- Apply the model

- E.g. predict



Learning: first look

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

2. Model selection:

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

E.g. $y = ax + b$

3. Choose the objective function (loss function)

- **Squared error**

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

4. Learning:

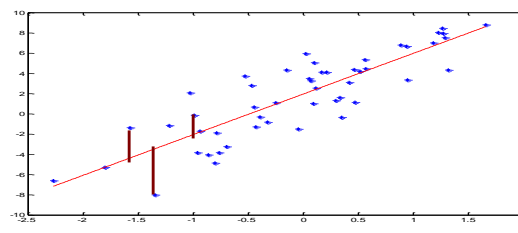
- **Find the set of parameters**

- The model and parameters with the smallest error

5. Application:

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

- E.g. predict y s for new inputs x



Learning: first look

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

2. Model selection:

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

E.g. $y = ax + b$

3. Choose the objective function (loss function)

- **Squared error**

4. Learning:

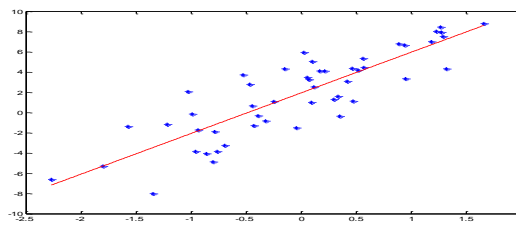
- **Find the set of parameters optimizing the error function**

- The model and parameters with the smallest error

5. Application:

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

- E.g. predict y s for new inputs x using learned $f(x)$



Learning: first look

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

2. Model selection:

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

E.g.

3. Choose the

- Squared error

4. Learning

- Find the

- The model

5. Application

- Apply the learned model to new data
- E.g. predict y s for new inputs \mathbf{x} using learned $f(\mathbf{x})$

