Learning

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:**
  - Concept learning, explanation-based learning, etc.
Supervised learning

Data: \( D = \{d_1, d_2, \ldots, d_n\} \) \hspace{1em} a set of \( n \) examples
\[ d_i = \langle x_i, y_i \rangle \]
\( x_i \) is input vector, and \( y \) is desired output (given by a teacher)

Objective: learn the mapping \( f : X \rightarrow Y \)
\[ s.t. \quad y_i \approx f(x_i) \quad \text{for all} \quad i = 1, \ldots, n \]

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 \( \rightarrow \) company stock price

• **Classification:** \( Y \) is **discrete**
  Handwritten digit (array of 0,1s) \( \rightarrow \) Label “3”
Unsupervised learning

- **Data:** \( D = \{d_1, d_2, \ldots, d_n\} \)
  \[ d_i = x_i \] 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, e.g. relations between the diseases, symptoms, lab tests etc.

---

Unsupervised learning example.

- **Density estimation.** We want to build the probability model of a population from which we draw samples \( d_i = x_i \)
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 \(y\)s for values of \(x\)
- We get the data what should we do?

Learning bias

- **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 bias

- Problem is easier when we make an assumption about the model, say, \( f(x) = ax + b \)
- Restriction to a linear model is an example of the learning bias

Bias provides the learner with some basis for choosing among possible representations of the function.

Forms of bias: constraints, restrictions, model preferences

Important: There is no learning without a bias!
Learning bias

- 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 \( a, b \)

![Graph showing data points and lines](image)

Fitting the data to the model

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

**How is the best set defined?**

Our goal is to have the parameters that:
- reduce the misfit between the model and data
- Or, (in other words) that explain the data the best

**Error function:**

**Gives a 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. \( y = ax + b \)

- **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 and not completely known population):

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

Expected squared error

The training error tries to approximate the true error.

But does a good training error always 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 mean-squares error
- Error is nonzero

![Graph showing linear function](image1)

**Overfitting**

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

![Graph showing cubic polynomial](image2)
Overfitting

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

• 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

- The 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)
Evaluation framework

- We want our classifier to generalize well to future examples
- **Problem:** But we do not know all future examples !!!
- **Solution:** evaluate the classifier on the **test set** that is withheld from the learning stage

![Diagram of Evaluation framework]

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
  \[ \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2 \]
- The generalization error is more objectively estimated using a separate test data set with \( m \) data samples
- **(Mean) test error**
  \[ \frac{1}{m} \sum_{j=1}^{m} (y_j - f(x_j))^2 \]
Design of a learning system

1. Data: \( D = \{d_1, d_2, \ldots, d_n\} \)

2. Model selection:
   - **Select a model** or a set of models (with parameters)
     E.g. \( y = ax + b \)
   - **Select the error function** to be optimized
     E.g. \( \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2 \)

3. Learning:
   - **Find the set of parameters optimizing the error function**
     \( \text{The model and parameters with the smallest error} \)

4. Application:
   - **Apply the learned model**
     \( \text{E.g. predict } y \text{ for new inputs } x \text{ using learned } f(x) \)