Learning

Vasileios Megalooikonomou

(some material adopted from notes by M. Hauskrecht)

Machine Learning

- The field of machine learning studies design of computer programs (agents) capable of learning from past experience and adapt to the new environment

- The need for building agents capable of learning is everywhere
  - Medical diagnosis, text classification, speech recognition, image/text retrieval, commercial software

- Machine learning is not only the deduction but induction of rules from examples that facilitate prediction and decision making
Learning

• Learning process:
  – Learner (a computer program) processes data $D$ representing the past experiences and tries to either derive something reasonable about the data seen or to develop some appropriate response to future data

• Example:
  – Learner sees a set of patient cases with corresponding disease labels and tries to predict the disease for future patient cases

Types of learning

Three main types:

• Supervised learning
  – Learning mapping between inputs $x$ and desired outputs $y$
  – Teacher provides $y$’s for the learning purposes

• Unsupervised learning
  – Learning relations between data components

• Reinforcement learning
  – Learning mapping between inputs $x$ and desired outputs $y$
  – Critic does not provide $y$’s but instead a signal (reinforcement) of how good my answer was
## Supervised learning

Data: \( D = \{ d_1, d_2, ..., d_n \} \) \text{ 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 (provided by a teacher)  

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

Two types of problems:  
- **Regression:** \( Y \) is continuous  
  Example: earnings, product orders \( \rightarrow \) company stock price  
- **Classification:** \( Y \) is discrete  
  Example: temperature, heart rate \( \rightarrow \) disease

## Unsupervised learning

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

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

Types of problems:  
- **Clustering**  
  Group together “similar” sample instances, e.g. patient cases  
- **Density estimation**  
  – Model probabilistically the population of samples
Unsupervised learning. 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

• Mixture of Gaussians – gives a probability distribution of a point in two dimensional space being seen.)
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 want to learn the mapping \( f : X \rightarrow Y \) for all possible 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 samples still unseen!

\[
\begin{align*}
\text{Problem is easier when we make an assumption about the model, say,} & \quad f(x) = ax + b + \epsilon \\
\epsilon & \sim N(0, \sigma) \quad \text{- random (normally distributed) noise} \\
\text{Restriction to the linear model is an example of learning bias}
\end{align*}
\]
Learning bias

- **Bias** provides the learner with some basis for choosing among possible representations of the function.
- **Bias**: constraints, restrictions, preferences among models
- **There is no learning without bias!**

Choosing a parametric model or a set of models is not enough. Still too many functions $f(x) = ax + b + \epsilon \quad \epsilon = N(0, \sigma)$

- 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 reduce the misfit between what model suggests and what data say
- Or, that explain the data the best

**Error function:**

**Measure of misfit between 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

![Graph showing linear regression](image_url)
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 other problems one must be careful about

– One of the most significant is **overfitting**

---

Overfitting

- Assume we have a set of 10 points and we consider polynomial functions as possible models
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 for 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? NO!!
• More important: How do we perform on the unseen data?
Generalization

- We would like the learner to predict correctly the values on the whole population of samples (many of them unseen in the training set)
- The true error of the learner is defined upon this population
- **Generalization error:** \( E_{(x,y)} (y - f(x, w))^2 \)  
  Expected squared error
- **(Mean) training error:** \( \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, w))^2 \)
- How well the training error approximates the true error?
- How to compute the generalization error? Approximation: Use separate data set with \( m \) data samples to test it
- **(Mean) test error** \( \frac{1}{m} \sum_{j=1}^{m} (y_j - f(x_j, w))^2 \)

Overfitting

- Situation when training error is low and generalization (test) 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)
Overfitting. Solutions.

- **Increase the number of samples**
  - May not be possible
- **Divide data set to a training set and validation set**
  - Train (fit) on the training set;
  - Check for the generalization error on the validation set, choose the model based on the validation set error.
- **Regularization (Occam’s Razor)**
  - Penalize for the model complexity (number of parameters)
  - Explicit preference towards simple models