How did we meet data and design algorithm in classic computer science and how do we now?

September 8, 2016

How did we meet data and design algorithm in classic computer science?

1. Sort
2. Matching
3. Shortest Path
4. SAT

No matter how the input data varies, we can always guarantee that our algorithm achieves our objective precisely.

How do we meet data and design algorithm now?

Using machine learning as an example

How do we meet data and design algorithm now?

For example, a classification problem:

- Given the features of some new customers of a bank, can we judge whether they will buy a financial product within a year, based on the experience.
- The experience means the data of the features of old customers and whether they bought any financial product during the first year they registered as a customer of the bank.

Table : A toy example of data

Feature #1	Feature #2	Feature #3	Tag
Age	Income	Also a customer of another bank	Brought or Not
28	10000	0 (No)	0 (No)
...	...	...	...
40	30000	1 (Yes)	1 (Yes)

How do we meet data and design algorithm now?

- **Problem**: In abstract level, we have a couple of samples (data) generated from the same data generator. Each sample consists of a feature vector $\vec{x}_i$ and a tag $y_i \in \{0, 1\}$. If there are some other samples generated by the data generator, can we predict the tags of these samples as precisely as possible by just knowing their feature vectors.
- **Objective**: What we really want to know is the map from the feature vector to the tag implied by the data generator.

Algorithm Design

To design a corresponding algorithm, we convert the objective to one of the two followings:

1. We want to find a map without lose of generalizability to best fit the existing samples, based on the belief that the map which best fits the existing sample will best act on the new generated data.
2. We want to find a map which will generate the existing data with highest probability, based on the assumption that the data generator follows certain type of stochastic mapping mechanism to generate the data.

The nature is an optimization problem!

Logistic Classification Algorithm

Suppose the data is generated (or the relation is) in the function form

$$y_i = h(\vec{x}_i) = \frac{1}{1 + e^{-\vec{\theta}^{\mathrm{T}} \vec{x}_i}}$$

where $\vec{\theta}$ are free variables to be determined. We aim to maximize the generating probability of the existing data

$$l(\vec{\theta}) = \Pi_i h(\vec{x}_i)^{y_i} (1 - h(\vec{x}_i))^{1-y_i}$$

This can be done by using classic optimization algorithm, the gradient descent.

1. Set $\vec{\theta}$ to arbitrary values
2. Update $\vec{\theta} = \vec{\theta} + \alpha \frac{dl}{d\vec{\theta}}$ till converge,
   or more commonly, update one sample by one sample,
   $\vec{\theta} = \vec{\theta} + \alpha(y_i - h(\vec{x}_i))\vec{x}$.

Logistic Classification

In sum,

- **Input**: labeled data samples as points in high dimension space with a classification tag 0 or 1
- **Objective**: find a linear function embedded in a logistic function that can predict the unlabeled data samples
- **Output**: the function parameters

How do we meet data and design algorithm now?

1. Supervised learning:
   - Support Vector Machine (SVM)
   - Neural networks (NN)
2. Unsupervised learning:
   - Principal Component Analysis (PCA)
   - K-means
3. Reinforcement learning

SVM

- **Input**: labelled data samples as points in a high dimension space
- **Objective**: find out a hyperplane in a high dimension that best separates samples with different tags
- **Output**: a hyperplane that separate two types of samples while maximizes the sum of the minimum distance to the hyperplane of samples of each type.

Neural Networks

- **Input**: labelled data samples as points in a high dimension space
- **Objective**: neural networks can mimic arbitrary function. Find a function without lose of generalizability that best fits the relation of existing data
- **Output**: a function in the form of a neural network that maximizes the fitness (generating probability) of the given data while keeps the generalizability

PCA

- **Input**: data samples as points in a high dimension space
- **Objective**: find out a few orthogonal directions such that the distances of these points on these directions are maximized
- **Output**: a couple of orthogonal directions

K-means

- **Input**: data samples as points in a high dimension space
- **Objective**: identify the similarity of the data and which pair of samples is more similar than another pair. The similarity is in a relatively vague meaning.
- **Output**: a couple of center points in a high dimension space that the sum of distance of each given point to the nearest center is maximized

Reinforcement Learning

- **Input**: an environment which consists of an action space, an state space and a state transition function and a reward function on states.
- **Objective**: find an action generator that can generate according to state transition history a sequence of actions that maximizes the final reward.