Machine Learning for Deep Learning

Sung-Yub Kim

Dept of IE, Seoul National University

January 14, 2017

Definition

Machine Learning Algorithm

"A computer program is said to learn from experience **E** *with respect to some class of tasks* **T** *and performance measure* **P** *if its performance at tasks in* **T**, *as measured by* **P**, *improves with experience* **E**."

Most of learning algorithms experience data

- Supervised Learning
  Each data contains **feature**, and each data is also associated with **label** or **target**
  ex) How much the used car is? Whether the price of asset go up?
- Unsupervised Learning
  Each data contains **features**, machine is asked to learn useful properties of the strucure of this dataset. The output of learning algorithm is usually distribution of data whether explicitly or implicitly.
  ex) Density estimation, Synthesis, Denoising

These two concepts are not formal

ML problem of one section can be transformed to the other section by

$$p(\mathbf{x}) = \prod_{i=2}^{n} p(x_1)p(x_i|x_1, \ldots, x_{i-1}) \tag{1}$$

and

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}, y)}{\sum_{y'} p(\mathbf{x}, y')} \tag{2}$$

And other learning pearadigms can be possible.

- semi-supervised
  Basically supervised, but have partially filled label.
  ex) Infer blanks, Recover corrupted data
- Reinforcement Learning
  Interact with environment, more formally RL is solving concepts of Partially informed Markov Decision Process.
  ex) AlphaGo

Design Matrix

In design matrix, examples are saved in rows, and features are saved in columns.

	Outlook	Temperature	Humidity	Wind	Play-Tennis
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes

To make design matrix, you need to make your data **vector** in same dimension.
For generality, sometimes we need to describe our data by a set of examples.

Machine Learning tasks are usually described in terms of how the machine learning system should process an **example**.
An example is a collection of **features** that have been quantitatively measured from some object or event.
Example can be represented as $\mathbf{x} \in \mathbb{R}^n$ where each entry x_i of the vector is feature.

Exmples of T

- Classification
  Put examples and select one class between 1,. . . ,k. More formally, we need to make function f

$$f : \mathbb{R}^n \mapsto 1, \ldots, k \tag{3}$$

- Regression
  Put examples and select one real value, More formally, we need to make function f

$$f : \mathbb{R}^n \mapsto \mathbb{R} \tag{4}$$

- Fill the missing inputs
  If some inputs are corrupted or removed, we need to choose whether we will make a model to fill all blanks or just estimate one probability distribution. First one is an Supervised Learning, and the second one is an Unsupervised Learning.

To evaluate the abilities of a ML algorithm, we must design a quantitative measure of its performance.

- If T is a classification problem or regression problem, then we need to define **error rate** which will be minimized.
  ex) 0-1 loss for classification, sum of squared error for regression
- For density estimation problem, since we cannot use error rate since there is no correct **label**, we need to define **cross entropy** $\mathbb{E}_p[-\log q]$ which will be maximized.

Since we are interested in how well the ML algorithm perform on data that it has not seen before, we evaluate performance measure using a **test data** which is separated before train algorithms.

ML algorithms need to perform well in *new, previously unseen* inputs not just those on model which we trained. We call this ability to perform well on unobserved inputs in **generalization**.

- Training Error
  When training a ML model, we have access to a training set; we can compute some error measure on the training set, called the **training error**; and we reduce this training error. In short, when training, we just solve minimization problem whose objective function is **training error**.
- Generalization Error
  When training process is over, we need to evaluate a ML model based on test set, and the measure of this evalutaion is called **Generalization Error**, also called **test error**.

In short, we need to minimize $\mathbb{E}_{p_{data}}[error]$, while we actually minimize in training. $\mathbb{E}_{p_{train}}[error]$ (In this context, p_{train} is empirical distribution)

If we assume that training set and test set are independent and idendically distributed(i.i.d assumption) and a model is independent from those two data sets, we can say that

$$E_{p_{data}}[error] = E_{p_{train}}[error] = E_{p_{test}}[error] \tag{5}$$

However, since we train model using train data, model is not independent. Therefore, we get

$$E_{p_{train}}[error] \leq E_{p_{test}}[error] \tag{6}$$

The factors determining how well a ML algorithm will perform are its ability to

1. Make the training error small(If not **underfitting** occurs)
2. Make the gap between training and test error small(If not **overfitting** occurs)

A model's capacity is its ability to fit a wide variety of functions. If capacity is too low, then hard to fit. If too high, then overfit problem may occurs.

- **Control of Capacity**

1. Choosing Hypothesis Spcae : Control a family of functions
2. Representational Capcity : Control range of the parameters

- **VC-dimenstion**
  VC-dimenstion measures the capcity of a binary classifier.

- **The most important theorem in Statitstical Learning**

$$\mathbb{P}[\textit{test error} \leq \textit{training error} + \sqrt{\frac{1}{N}[D(\log(\frac{2N}{D}+1) - \log(\frac{\eta}{4}))]}] = 1 - \eta \tag{7}$$

where D is VC-dimension of model, N is the number of data Therefore, we get **sample-complexity bounds** like

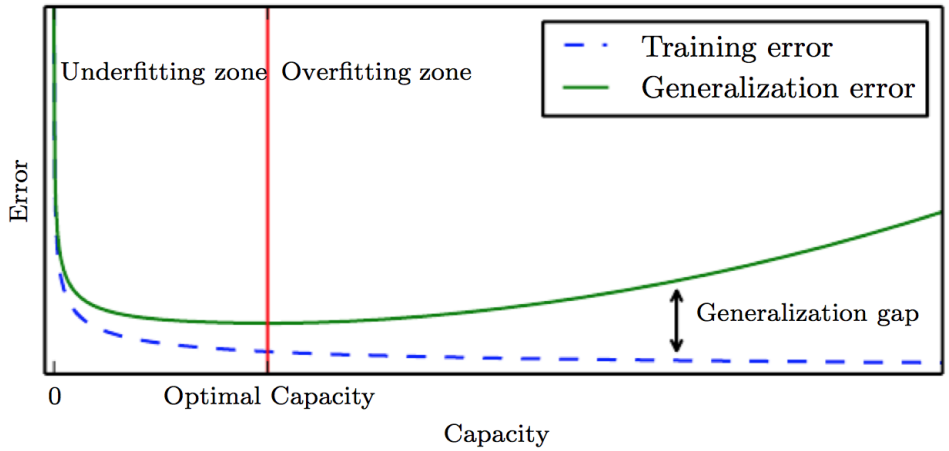
$$N = \Theta(\frac{D + \ln\frac{1}{\delta}}{\epsilon}) \tag{8}$$

But this complexity cannot be adopted to deep learning (TOO BIG)

- **Bayes Error**

The error incurred by an oracle making prediction from the true distribution $p(\mathbf{x}, y)$ is called the **Bayes error**.

- Relationship between Capcity and Error



If we have enough capacity and data, then by VC-dimenstion $E_{p_{train}}[error] \simeq E_{p_{test}}[error]$ and generalization error converges to bayes error. But if we don't have enough capacity, these phenomena would not occur.

- **No Free Lunch Theorem for ML**

"Averaged over all possible data-generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points."
This theorem argue that no ML algorithm is universally any better than any other.
Therefore, the goal of ML is not to seek a universal learning algorithm or the absolute learning algorithm. Instead, our goal is to understand what kinds of distributions are relevant to the 'Real World', and what kinds of ML algorithms perform well on data drawn from the kinds of data-generating distributions we care about.

Instead we explicitly remove some set of functions, we can get model which is more generalized by adding regularizer like $\Omega(\mathbf{w}) = \mathbf{w}^\top \mathbf{w}$.
This type of regularization can be occured, since Lagrangian of this optimization problem

$$\min_{\mathbf{w}} \{E_{p_{train}}[error] : \|\mathbf{w}\| \leq \alpha\} \tag{9}$$

is

$$E_{p_{train}}[error] + \lambda \|\mathbf{w}\|^2 \tag{10}$$

and the difference between solving original problem and minimizing Lagrangian is just we don't know which λ is the best exactly.
Also there exist many other regularization technique which can be used implicitly.

- **Hyper-parameter**

Hyper-parameter is a parameter that is given when ML model is learning. Sometimes we can optimize or learn it, but most of time ir is not appropriate to learn that hyper-parameter on the training set. This applies to all hyper-parameters that control model capacity.

- **Validation Set**

To solve above problem, we need a validation set of example that the training algorithm does note observe. Also, examples in validation set cannot be included in test set also, since this **optimizing hyper-parameter** process is also part of learning. In short, if we optimize a hyper-parameter we train model twice. The ratio of train data to validation data is 8:2 usually.

- **K-fold Cross Validation**

If you don't have enough data, previous dividing data proocess would be critical to performance of model. In that case, you need to average your model evaluation by repeating devision process of train-test. That means you need to divide data for k partitions and you need to train k times using (k-1) partitions of data and test using the remained partition. Then average your model performance to get a estimation of real model performance. One problem is that no unbiased estimators of the variance of such average error estimators extst.