Introduction to Machine Learning

Shashank Srikanth

IIIT Hyderabad

May 11, 2020

Overview

What is machine learning (3)

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.

Tom Mitchell, ML Professor at CMU

- It is a process of "automating automation"
- It has applications in several fields ranging from:
  - Email spam detection
  - Object detection
  - Drug prediction
  - Stock prediction . . .
- Simply put - ‘Machine learning is the hot new thing‘

ML Paradigm

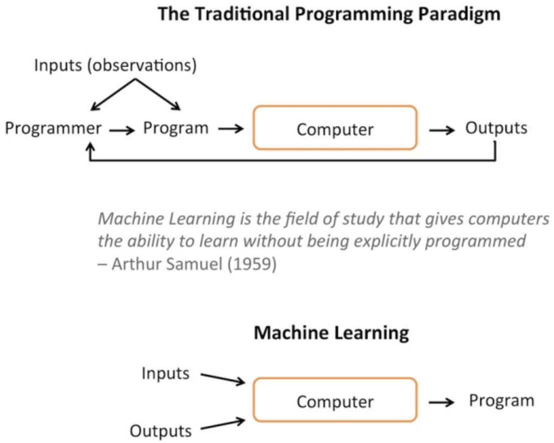

The Traditional Programming Paradigm
Inputs (observations)
Programmer
Program
Computer
Outputs
Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed
– Arthur Samuel (1959)
Machine Learning
Inputs
Outputs
Computer
Program

Figure: Machine learning vs. “classic” programming

Categories of ML

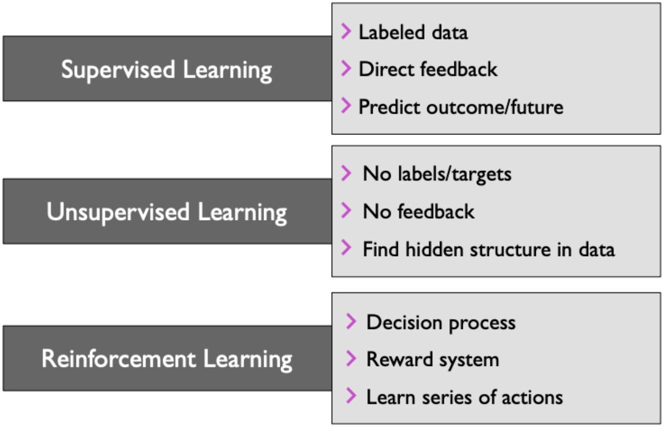


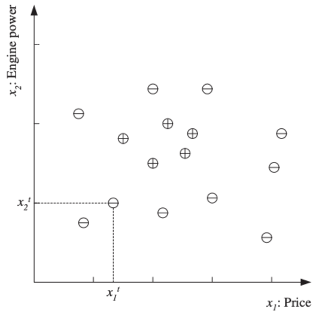
Figure: Categories of Machine Learning

Supervised Learning

- Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples.
- It usually focuses on learning a classification or regression model
- For example, we are given a training set of labeled examples (positive and negative) and want to learn a classifier that we can use to predict unseen examples, or to understand the data

Supervised Learning

training set for a "family car"



Hypothesis class of rectangles

$(p_1 \leq \text{price} \leq p_2)$ AND $(e_1 \leq \text{engine power} \leq e_2)$

where $p_1, p_2, e_1, e_2 \in \mathbb{R}$

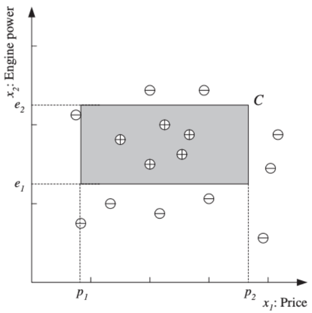


Figure: Illustration of binary classification problem

Unsupervised Learning

- In contrast to supervised learning, unsupervised learning is a branch of machine learning that is concerned with unlabeled data
- Common tasks in unsupervised learning are clustering analysis (assigning group memberships) and dimensionality reduction (compressing data onto a lower-dimensional subspace).

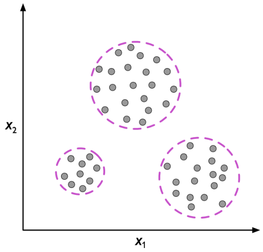


Figure: Illustration of clustering

Reinforcement Learning

- Reinforcement is the process of learning from rewards while performing a series of actions.
- In reinforcement learning, we do not tell the learner or agent, for example, a (ro)bot, which action to take but merely assign a reward to each action and/or the overall outcome. Instead of having "correct/false" label for each step, the learner must discover or learn a behavior that maximizes the reward for a series of actions

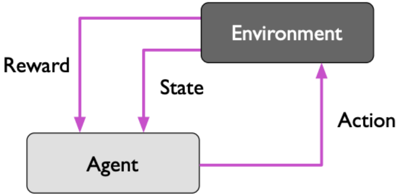


Figure: Illustration of reinforcement learning

Introduction to Supervised Learning

- We are given a labeled training dataset from which a ML algorithm learns a model that predicts labels of unlabeled data points
- We define h as the "hypothesis," a function that we use to approximate some unknown function $f(x) = y$, where x is a vector of input features (training example) and y is the outcome we want to predict

Classification

In classification, the hypothesis function is defined as $h : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{X} \in \mathbb{R}^m$ and $\mathcal{Y} \in \{0, 1, \ldots, k\}$. Example - cats vs dogs classifier

Regression

In regression, the hypothesis function is defined as $h : \mathbb{R}^m \rightarrow \mathbb{R}^k$. Regression usually maps an input to continuous output variables (y). Example - Linear regression / Stock value prediction

Overview of ML Pipeline

Training set

- Given a training set $\mathcal{D} = \{< \mathbf{x}^{[i]}, y^{[i]} >, i = 1, \ldots, n\}$, we denote the ith training sample as $< \mathbf{x}^{[i]}, y^{[i]} >$
- A critical assumption for (most) machine learning theory is that the training examples are i.i.d. (independent and identically distributed).

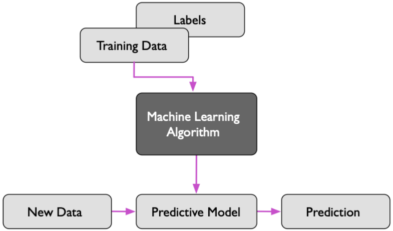


Figure: Rough overview of the supervised learning process.

Hypothesis Space

- Machine learning algorithms sample from a hypothesis space that is usually smaller than the entire space of all possible hypotheses $\mathcal{H}$ – an exhaustive search covering all $h \in \mathcal{H}$ would be computationally infeasible since H grows exponentially with the size (dimensionality) of the training set.
- ML is often used to reduce the hypothesis space - For example, a neural network with a single hidden layer, a finite number of neurons, and non-linear activation functions such as sigmoid units, was proved to be a universal function approximator

Algorithm Categories

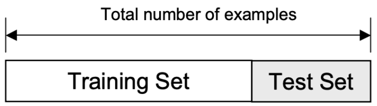
- **Batch vs online learning**: Batch learning refers to the fact that the model is learned on the entire set of training examples. Online learners, in contrast, learn from one training example at the time
- **Parametric vs Nonparametric models**
  - Parametric models are "fixed" models, where we assume a certain functional form for $f(\mathbf{x}) = y$. For example, linear regression can be considered as a parametric model with $h(\mathbf{x}) = w_1 x_1 + \ldots + w_m x_m + b$.
  - Nonparametric models do not have a pre-specfied number of parameters. For example, in a decision tree, each decision node (e.g., a binary "True/False" assertion) can be regarded as a parameter.
- **Discriminative vs generative**:
  - Generative models (classically) describe methods that model the joint distribution $P(X, Y) = P(Y)P(X|Y) = P(X)P(Y|X)$ for training pairs $< \mathbf{x}^{[i]}, y^{[i]} >$.
  - Discriminative models are taking a more "direct" approach for modeling $P(Y|X)$ directly.

Model selection

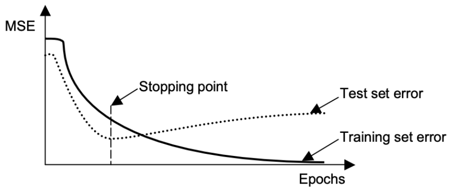
- Validation techniques are motivated by two fundamental problems in pattern recognition: model selection and performance estimation
    - **Model Selection**: How do we select the "optimal" parameter(s) or model for a given classification/regression problem?
    - **Performance estimation**: Once we have chosen a model, how do we estimate its performance?
- In real applications we only have access to a finite set of examples, usually smaller than we wanted and we would like to reduce the error in production
- If we use the whole dataset for training - we might overfit on this data and the model may not generalize well to unseen data
- TRUE ERROR RATE: The classifier's error rate on the ENTIRE POPULATION

Holdout method

- Split dataset into two groups
    - **Training set**: Used to train the classifier
    - **Testing set**: Used to estimate the error rate of the trained classifier

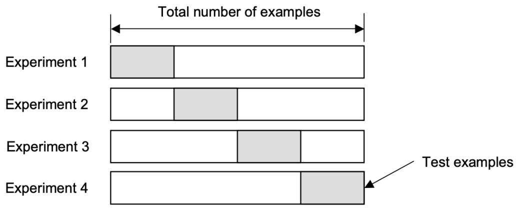


- Often, the holdout/test set is used to determine the stopping point of the training procedure



K-Fold Cross Validation

- We create a K-fold partition of the dataset
  - For each of K experiments, use K-1 folds for training and the remaining one for testing



- The true error is estimated as the average error rate

$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

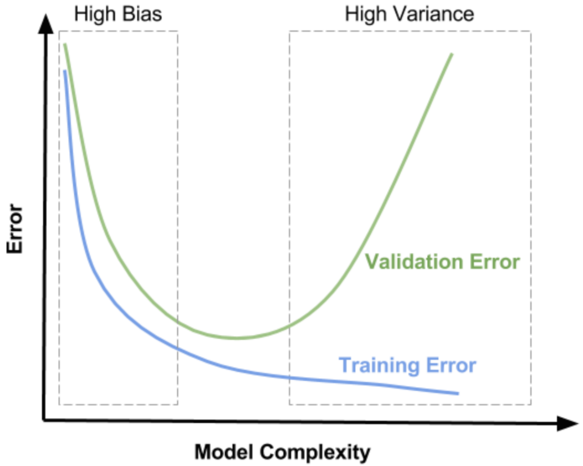
No of folds

- With a large number of folds
    - The bias of the true error rate estimator will be small (the estimator will be very accurate)
    - The variance of the true error rate estimator will be large
    - The computational time will be very large as well (many experiments)
- With a small number of folds
    - The number of experiments and, therefore, computation time are reduced
    - The variance of the estimator will be small
    - The bias of the estimator will be large (conservative or higher than the true error rate
- For large datasets, even 3-Fold Cross Validation will be quite accurate
- For very sparse datasets, we may have to use leave-one-out in order to train on as many examples as possible

Bias Variance Tradeoff

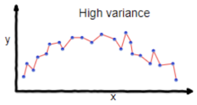
- Total Error = Bias + Variance + Irreducible Error
- Even a perfect model will not be able to reduce all the errors as the training data itself may contain noise. Thus, we try to reduce the bias and variance as much as possible
- **Bias**: Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very oversimplifies the model and this leads to high error on training and test data.
- **Variance**: Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. A model with high variance do not generalize on the data which it hasn't seen before. Such models perform very well on training data but have high error rates on test data.

Bias Variance Tradeoff

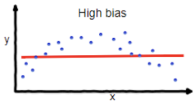


Overfitting & Underfitting

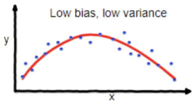
- **Underfitting** happens when a model unable to capture the underlying pattern of the data. These models usually have high bias and low variance. It happens when we have very less amount of data to build an accurate model or when we try to build a linear model with a nonlinear data.
- **Overfitting** happens when our model captures the noise along with the underlying pattern in data. It happens when we train our model a lot over noisy dataset. These models have low bias and high variance



overfitting



underfitting



Good balance

Maximum Likelihood Estimation (4)

- Maximum Likelihood Estimation (MLE) and Maximum A Posteriori (MAP), are both a method for estimating some variable in the setting of probability distributions or graphical models.
- In **MLE**, we choose θ that maximizes the probability of observed data.
- When fitting a Gaussian to our dataset, we immediately take the sample mean and sample variance, and use it as the parameter of our Gaussian.
- **Why do we care about this?** Most of the optimization in Machine Learning and Deep Learning (neural net, etc), could be interpreted as MLE.

Maximum Likelihood Estimation

- Let the likelihood function be $P(X|\theta)$. Then, the MLE for θ is

$$\begin{aligned}\theta_{MLE} &= \arg\max_{\theta} P(X|\theta) \\ &= \arg\max_{\theta} \prod_i P(x_i|\theta)\end{aligned}$$

- Above product approaches 0 as the number of those numbers goes to infinity. Hence, we will instead work in the log space, as logarithm is monotonically increasing, so maximizing a function is equal to maximizing the log of that function.

$$\begin{aligned}\theta_{MLE} &= \arg\max_{\theta} \log P(X|\theta) \\ &= \arg\max_{\theta} \log \prod_i P(x_i|\theta) \\ &= \arg\max_{\theta} \sum_i \log P(x_i|\theta)\end{aligned}$$

MLE for Gaussian

- The parameters of a Gaussian distribution are the mean (μ) and variance (σ^2). Given observations $x_1, \ldots, x_N$, the likelihood of those observations for a certain μ and σ^2 (assuming that the observations came from a Gaussian distribution) is

$$p\left(x_1, \ldots, x_N \middle| \mu, \sigma^2\right) = \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ \frac{-\left(x_n - \mu\right)^2}{2\sigma^2} \right\}$$

- The log likelihood for the same is given below. We can take the derivative of the log likelihood wrt the parameters to get the MLE parameters for a gaussian, which is μ and σ itself. But the sigma that we get from MLE is a biased estimator

$$\mathcal{L}(\mu, \sigma) = -\frac{1}{2} N \log\left(2\pi\sigma^2\right) - \sum_{n=1}^{N} \frac{\left(x_n - \mu\right)^2}{2\sigma^2} \tag{1}$$

Maximum A Posteriori (MAP)

- MAP usually comes up in Bayesian setting. Because, as the name suggests, it works on a posterior distribution, not only the likelihood. With Bayes' rule, we could get the posterior as a product of likelihood and prior:

$$\begin{aligned} P(\theta|X) &= \frac{P(X|\theta)P(\theta)}{P(X)} \\ &\propto P(X|\theta)P(\theta) \end{aligned}$$

- We are ignoring the normalizing constant as we are strictly speaking about optimization here, so proportionality is sufficient.
- Often, the prior is based on expert knowledge. We can have several different forms of prior such as:
    - Uniform distribution
    - Normal distribution...

Maximum A Posteriori (MAP)

$$
\begin{aligned}
\theta_{MAP} &= \arg\max_{\theta} P(X|\theta)P(\theta) \\
&= \arg\max_{\theta} \log P(X|\theta) + \log P(\theta) \\
&= \arg\max_{\theta} \log \prod_{i} P\left(x_i|\theta\right) + \log P(\theta) \\
&= \arg\max_{\theta} \sum_{i} \log P\left(x_i|\theta\right) + \log P(\theta)
\end{aligned}
$$

- Comparing both MLE and MAP equation, the only thing differs is the inclusion of prior $P(\theta)$ in MAP, otherwise they are identical. For uniform prior, we assign equal weights everywhere, on all possible values of the θ.
- On deriving the MAP in this case, we find that it is equal to MLE. Thus, MLE is a special case of MAP, where the prior is uniform!

References

[1] MiguelA Carreira-Perpinán. Cse176 - introduction to machine learning—lecture notes. 2016.

[2] Ricardo Gutierrez-Osuna. Lecture 13: Validation. `http://research.cs.tamu.edu/prism/lectures/iss/iss_l13.pdf`.

[3] Sebastian Raschka. Stat 479: Machine learning lecture notes, 2018.

[4] Agustinus Kristiadi. Mle vs map: the connection between maximum likelihood and maximum a posteriori estimation. `https://wiseodd.github.io/techblog/2017/01/01/mle-vs-map/`.