Introduction to ML

Eduardo Morales, Hugo Jair Escalante

INAOE

Introduction

- Machine Learning tries to build programs that automatically improve their performance with experience
- Learning is perhaps the most distinctive characteristic of human intelligence
- Since the beginning of computing, researchers asked themselves whether machines could be able to learn: “”what we want is a machine that can learn from experience” (A. Turing, 1947)
- Understanding how machines learn may help us to understand human learning

Introduction

- In order to solve problems we create programs/models
- Some tasks are difficult to formalize, there are not available experts, there might be too much data, $\ldots \Rightarrow$ ML
- ML automatically generates programs/models from data
- This opens an almost endless possible applications

Taxonomy

Authors classify ML approaches from different perspectives:

- Underlying mathematical model
- Nature of the data
- Task being solved
- Suppositions on the model

Underlying Mathematical Model

1. **Geometric Models**: The examples define a space of instances where geometric models can be build, e.g., evaluate distances, search for hyper-planes, find prototypes, etc.

- Usually the attributes are numeric so it is easy to use geometric concepts like lines, planes, and distances, and to do linear transformations and apply different distance measures
- Some ML examples are: Linear classifiers, k-nearest neighbors, k-means, clustering in general, SVMs, kernel classifiers, classifiers based on prototypes, etc.

Underlying Mathematical Model

2. **Probabilistic Models**: In ML we want to find the best (most probable) hypothesis given the data

- If $P(D)$ = *a priori* probability of the data (i.e., which data is more probable than other) and $P(D \mid h)$ = probability of the data given the hypothesis, what we want to estimate is: $P(h \mid D)$, the *posterior* probability of h given the data
- This can be estimated using Bayes rule:

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)}$$

Underlying Mathematical Model

- To estimate the most probable hypothesis or MAP (*maximum a posteriori hypothesis*):

$$\begin{aligned} h_{MAP} &= argmax_{h \in H} \left(P(h \mid D) \right) \\ &= argmax_{h \in H} \left(\frac{P(D|h)P(h)}{P(D)} \right) \\ &\approx argmax_{h \in H} \left(P(D \mid h)P(h) \right) \end{aligned}$$

  since $P(D)$ is constant and independent of h.
- If we suppose that all the hypotheses are equally likely, we end-up with the *maximum likelihood* hypothesis:

$$h_{ML} = argmax_{h \in H} \left(P(D \mid h) \right)$$

- Under this mathematical framework it is common to use concepts such as *a priori* and *posterior* probabilities, *maximum likelihood*, Bayes theorem, etc.

Underlying Mathematical Model

3. **Logical Models**: Models that can be expressed using logic, which includes conjunctions, disjunctions, negation, etc.

- They are also known as declarative models and can be used to provide explanations
- It is common to use logic concepts like *complete* and *consistent*
- ML examples are classification rules, decision trees, ILP, frequent patterns, subgroup discovery, etc.

Nature of the Data

This is perhaps the most common ML classification:

1. **Supervised Learning**: There is X data associated with a label (class) Y and the goal is to find a model that given an instance of X predicts a label in Y. This includes classification and regression tasks, and a commonly used concept is *over-fitting*
2. **Unsupervised Learning**: In this case there are no associated labels and the goal is to find an inherent structure in the data to organize it by similarity or relations among variables
3. **Reinforcement Learning**: Learns how to map states to actions in order to solve a sequential decision problem, through an iterative process of exploring the environment

ML Tasks

- **Description**: Obtains descriptions of the data, produces summaries, find prototypical examples, etc.
- **Prediction**: Performs classification (discrete labels) and estimation or regression (continuous labels) tasks
- **Segmentation**: Divides the data into groups or clusters
- **Dependency analysis**: Finds dependencies among variables and their values
- **Anomaly detection** and extreme cases
- **Control**: Learns which action to take at each state
- **Optimization and search**: Not always considered ML

Suppositions over the models

- **Parametric**: The model summarizes the data with a finite set of parameters
- **Non parametric**: There are no strong suppositions in terms of the function or model that is induced

Parametric Models

These algorithms follow two steps:

1. Select the function
2. Learn the values of the coefficients of the function from data

Parametric Models

A large set of functions can be used, for instance:

- Linear functions
- Logistic regression
- Perceptrons
- Naïve Bayes
- Simple neural networks
- ...

Parametric Models

Advantages:

- Simple: Easy to understand and interpret their results
- Speed: They are learned quickly
- Data: In general needs fewer data

Disadvantages:

- Restrictive: The selected function constrains what can be learned
- Limited Complexity: There are adequate, in general, to simpler problems
- Fitting: It is possible that the selected model does not properly fit the underlying function

Non Parametric Models

- Make no assumptions on the shape of the function, which is determined by the data
- Some examples are:
  - k-nearest neighbors
  - Decision trees
  - SVM
  - Bayesian learning
  - ...

Non Parametric Models

Advantages:

- Flexible: Can create a wide range of functions
- Power: They do not make strong assumptions over the models
- Performance: Tend to obtain better performance

Disadvantages:

- Data: Require large quantities of data
- Speed: Can take longer to learn
- Adjustments: They are prone to overfit

Parametric Models

- Let us suppose that we know the distribution (e.g., Gaussian)
- The advantages of the parametric models is that they tend to depend on few parameters (e.g., mean and variance)
- To estimate the parameters we can use *maximum likelihood*

Maximum Likelihood Estimate

- Suppose an i.i.d. (independent and identically distributed) sample set $X = \{x_t\}_{t=1}^{N}$
- Suppose x_t is an instance taken from a family of known distributions, $p(x|\Theta)$, defined by parameters Θ
- What we want to estimate is Θ such that it is most probable to sample x from $p(x|\Theta)$

Maximum Likelihood Estimate

- Since x_t is independent, the likelihood of the parameters given X, can be evaluated by the product of the individual likelihoods:

$$l(\Theta|X) \equiv p(X|\Theta) = \prod_{t=1}^{N} p(x_t|\Theta)$$

- To obtain the maximum likelihood, we can take the logarithm and change the product into a summation:

$$\mathcal{L}(\Theta|X) \equiv \log l(X|\Theta) = \sum_{t=1}^{N} \log p(x_t|\Theta)$$

- For two classes we can use a Bernoulli distribution; for N classes we can use a multimodal Gaussian distribution

Bernoulli Distribution

- The probability of an event to occur ($X = 1$) is p and that the event will not occur ($X = 0$) is $1 - p$

$$P(x) = p^x(1-p)^{1-x}, x \in \{0,1\}$$

- The expected value and variance of X are:

$$E[X] = \sum_x xp(x) = 1 \cdot p + 0 \cdot (1-p) = p$$

$$Var(X) = \sum_x (x - E[X])^2 p(x) = p(1-p)$$

Bernoulli Distribution

- We have only one parameter p, and we want to estimate its value $\hat{p}$
- The *log likelihood* is:

$$\mathcal{L}(p|X) = \log \prod_{t=1}^{N} p^{x_t}(1-p)^{1-x_t}$$

$$= \sum_t x_t \log p + (N - \sum_t x_t) \log(1-p)$$

- We take the derivative with respect to p to maximize, $d\mathcal{L}/dp = 0$

$$\hat{p} = \frac{\sum_t x_t}{N}$$

- Which is what it is expected

Normal Distribution

- $Var(x) \equiv \sigma^2, E[X] = \mu$

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

- The *log likelihood* is:

$$\mathcal{L}(\mu, \sigma|X) = -\frac{N}{2}\log(2\pi) - N\log\sigma - \frac{\sum_t (x_t - \mu)^2}{2\sigma^2}$$

- Taking the partial derivatives of each argument and making them equal to zero:

$$m = \frac{\sum_t x_t}{N}$$

$$s^2 = \frac{\sum_t (x_t - m)^2}{N}$$

Parametric Classification

- Following a Bayesian approach:

$$p(C_i|x) = \frac{p(x|C_i)p(C_i)}{p(x)}$$

- Since the denominator is constant, the discrimination function is:

$$g_i(x) = p(x|C_i)p(C_i)$$

- Or equivalently:

$$g_i(x) = \log p(x|C_i) + \log p(C_i)$$

Parametric Classification

- If we suppose that $p(x|C_i)$ is Gaussian:

$$p(x|C_i) = \frac{1}{\sqrt{2\pi}\sigma_i} exp\left[-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right]$$

- The discrimination function is:

$$g_i(x) = -\frac{1}{2} log(2\pi) - log\sigma_i - \frac{(x-\mu_i)^2}{2\sigma_i^2} + logp(C_i)$$

- We can estimate the mean, standard deviation, and $P(C_i)$ from the data, and substitute:

$$g_i(x) = -\frac{1}{2} log(2\pi) - logs_i - \frac{(x-m_i)^2}{2s_i^2} + logp(C_i)$$

Parametric Classification

- The first term is constant, and if we assume that the probabilities of the classes and the variance are constant:
$$g_i(x) = -(x - m_i)^2$$
- We can then assign the class to the element that is closer to its mean:
- C_i si $|x - m_i| = min_k|x - m_k|$
- With two classes, the midpoint is the decision threshold: $g_1(x) = g_2(x)$
$$(x - m_i)^2 = (x - m_2)^2$$
$$x = \frac{m_1 + m_2}{2}$$

Regression

- In regression the output or dependent variable is a function of the inputs or independent variables: $r = f(x) + \epsilon$
- $f(x)$ is an unknown function that we want to estimate with $g(x|\Theta)$, defined by a set of parameters Θ
- If we suppose that ϵ is Gaussian noise with a zero mean and constant variance ($\epsilon \sim \mathcal{N}(0, \sigma^2)$), and placing our estimator $g(\cdot)$ instead of $f(\cdot)$:

$$p(r|x) \sim \mathcal{N}(g(x|\Theta), \sigma^2)$$

Regression

- Again, we want to find the parameters Θ with maximum likelihood
- We have a set of data (x, r) coming from certain density distribution $p(x, r)$, that we can write as:

$$p(x, r) = p(r|x)p(x)$$

- and the logarithm of its likelihood is:

$$\mathcal{L}(\Theta|X) = \log \prod_{t=1}^{N} p(x_t, r_t)$$

$$= \log \prod_{t=1}^{N} p(r_t|x_t) + \log \prod_{t=1}^{N} p(x_t)$$

Regression

- We can ignore the second term that does not depend on our estimator:

$$\mathcal{L}(\Theta|X) = \log \prod_{t=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} exp\left[-\frac{(r_t - g(x_t|\Theta))^2}{2\sigma^2}\right]$$

$$= \log\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^N exp\left[-\frac{1}{2\sigma^2}\sum_{t=1}^{N}(r_t - g(x_t|\Theta))^2\right]$$

$$= -N\log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2}\sum_{t=1}^{N}(r_t - g(x_t|\Theta))^2$$

- The first term, as well as $1/\sigma^2$, are independent of the parameters, which reduces to the most commonly used loss function:

$$= -\frac{1}{2}\sum_{t=1}^{N}(r_t - g(x_t|\Theta))^2$$

Regression

- For intances, in linear regression, we have: $g(x_t|w_1, w_0) = w_1 x_t + w_0$
- If we take the derivative of the loss function with respect to w_1 and w_0:

$$\sum_t r_t = N w_0 + w_1 \sum_t x_t$$

$$\sum_t r_t x_t = w_0 \sum_t x_t + w_1 \sum_t (x_t)^2$$

- Which can be rewritten in a matrix form as: $Aw = y$ where:

$$A = \begin{bmatrix} N & \sum_t x_t \\ \sum_t x_t & \sum_t (x_t)^2 \end{bmatrix}$$

$$w = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}; y = \begin{bmatrix} \sum_t r_t \\ \sum_t r_t x_t \end{bmatrix}$$

Regression

- Which can be solved as: $w = A^{-1}y$
- The same procedure can be extended to polynomial regressions and framing them as: $Ax = y$
- Roughly the same is done in multivariate problems
- The order of the polynomials is important. Higher order polynomials have more variance with small changes in the data, but can fit better the data
- There is a tradeoff between bias and variance

Cross Entropy

- Another common application is to learn a probability function with two possible outcomes (e.g., 0 o 1)
- If the data (D) are: $D = \{(x_1, d_1), \ldots, (x_m, d_m)\}$, where d_i is the observed value (0 or 1) of $f(x_i)$, and assuming that the data elements are independent:

$$P(D \mid h) = \prod_{i=1}^{m} P(x_i, d_i \mid h)$$

- If x_i is independent of h

$$P(D \mid h) = \prod_{i=1}^{m} P(d_i \mid h, x_i) P(x_i)$$

Cross Entropy

- Since *h* is the probability of the target function $P(d_i = 1 \mid h, x_i) = h(x_i)$, and in general:

$$P(d_i \mid h, x_i) = \begin{cases} h(x_i) & \text{if } d_i = 1 \\ 1 - h(x_i) & \text{if } d_i = 0 \end{cases}$$

- This can be written as:

$$P(d_i \mid h, x_i) = h(x_i)^{d_i}(1 - h(x_i))^{1-d_i}$$

- So:

$$P(D \mid h) = \prod_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} P(x_i)$$

Cross Entropy

- The maximum likelihood is then:

$$h_{ML} = argmax_{h \in H} \left(\prod_{i=1}^{m} h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} P(x_i) \right)$$

- Ignoring the last term (that does not depend on *h*), we have:

$$h_{ML} = argmax_{h \in H} \left(\prod_{i=1}^{m} h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} \right)$$

- Which is a generalization of the binomial distribution (e.g., describes the probability of $(d_1, \ldots, d_m)$ results when tossing m coins, assuming that each coin has a probability of $h(x_i)$ of being heads)

Cross Entropy

- In the Binomial distribution it is assumed that all coins have the same probability of heads
- Taking (again) the logarithm:

$$h_{ML} = argmax_{h \in H} \left(\sum_{i=1}^{m} d_i ln(h(x_i)) + (1 - d_i) ln(1 - h(x_i)) \right)$$

- Which due to its similarity to the entropy measure, its negative is called *cross entropy*.

Loss Functions

- The loss functions are used to optimize a model (minimize its loss)
- The two most commonly used loss functions in ML are:
  - Mean square error (MSE o L2): Regression

$$MSE = -\frac{1}{2}\sum_{t=1}^{N}(y_i - \hat{y}_i)^2$$

  - Cross Entropy: Classification
    For 2 classes:

$$CE(y,p) = -y\log(p) - (1-y)\log(1-p)$$

    For *m* classes:

$$CE(y,p) = -\sum_{c=1}^{m} y_c log(p_, c)$$

- Other loss functions include: Hinge, Huber, Kullback-Leibler, RMSE, MAE (L1)