Machine Learning

Edoardo Righi

October 13, 2019

1 Introduction

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

E.T. Mitchell

Successful applications of Machine Learning:

- Speech recognition, Optical character recognition, Computer Vision
- Learning to drive an autonomous vehicle (DARPA Grand Challenges, Google Self-Driving Car)
- Game playing (IBM's Deep Blue, Watson, AlphaGO)
- Recommender Systems

At the moment, big players are heavily investing in machine learning: Google, Facebook, Amazon, IBM, Uber, ...
To make it effective, the components of a learning problem have to be well-posed:

- **task**: to be addressed by the system (e.g. recognizing handwritten characters)
- **performance measure**: to evalute the learned system (e.g. number of misclassified characters)
- **training experience**: to train the learning system (e.g. labelled handwritten characters)

1.1 Designing a machine learning system

1. Formalize the learning task
2. Collect data
3. Extract features
4. Choose class of learning models
5. Train model
6. Evaluate model

1.2 Formalize the learning task

- Define the task that should be addressed by the learning system (e.g. recognizing handwritten characters from images)
- A learning problem is often composed of a number of related tasks. E.g.:
  - Segment the image into words and each word into characters.
  - Identify the language of the writing.
  - Classify each character into the language alphabet.
- Choose an appropriate performance measure for evaluating the learned system (e.g. number of misclassified characters)

1.3 Collect data

A set of training examples need to be collected in machine readable format. Data collection is often the most cumbersome part of the process, implying manual intervention especially in labelling examples for supervised learning. Recent approaches to the problem of data labeling try to make use of the much cheaper availability of unlabeled data (semi-supervised learning).

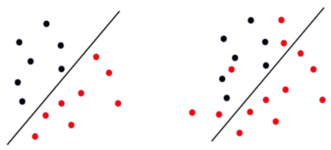
1.4 Extract features

A relevant set of features need to be extracted from the data in order to provide inputs to the learning system. Prior knowledge is usually necessary in order to choose the appropriate features for the task at hand.

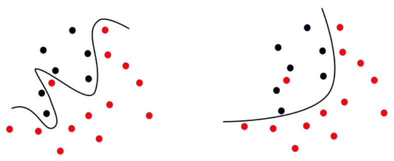
- Too few features can miss relevant information preventing the system from learning the task with reasonable performance.
- Including noisy features can make the learning problem harder.
- Too many features can require a number of examples greater than those available for training.

1.5 Choose learning model class

- A simple model like a linear classifier is easy to train but insufficient for non linearly separable data.



- A too complex model can memorize noise in training data failing to generalize to new examples.



1.6 Train model

Training a model implies searching through the space of possible models (aka hypotheses) given the chosen model class. Such search typically aims at fitting the available training examples well according to the chosen performance measure. However, the learned model should perform well on unseen data (generalization), and not simply memorize training examples (overfitting). Different techniques can be used to improve generalization, usually by trading off model complexity with training set fitting.

Entities are not to be multiplied without necessity

William of Occam (Occam's razor)

1.7 Evaluate model

The learned model is evaluated according to its ability to generalize to unseen examples.

- Evaluation can provide insights into the model weaknesses and suggest directions for refining/modifying it.
- Evaluation can imply comparing different models/learners in order to decide the best performing one

Statistical significance of observed differences between performance of different models should be assessed with appropriate statistical tests.

1.8 Learning settings

1.8.1 Supervised learning

- The learner is provided with a set of input/output pairs $(x_i, y_i) \in X \times Y$
- The learned model f : $X \to Y$ should map input examples into their outputs (e.g. classify character images into the character alphabet)
- A domain expert is typically involved in labeling input examples with the corresponding outputs.

1.8.2 Unsupervised learning

- The learner is provided with a set of input examples $x_i \in X$, with no labeling information
- The learner models training examples, e.g. by grouping them into clusters according to their similarity

1.8.3 Semi-supervised learning

- The learner is provided with a set of input/output pairs $(x_i, y_i) \in X \times Y$
- A (typically much bigger) additional set of unlabeled examples $x_i \in X$ is also provided.
- Like in supervised learning, the learned model $f : X \to Y$ should map input examples into their outputs
- Unlabelled data can be exploited to improve performance, e.g. by forcing the model to produce similar outputs for similar inputs, or by allowing to learn a better internal representation of examples.

1.8.4 Reinforcement learning

The learner is provided a set of possible states S, and for each state, a set of possible actions, A moving it to a next state. In performing action a from state s, the learner is provided an immediate reward $r(s, a)$. The task is to learn a policy allowing to choose for each state s the action a maximizing the overall reward (including future moves). The learner has to deal with problems of *delayed reward* coming from future moves, and trade-off between *exploitation* and *exploration*. Typical applications include moving policies for robots and sequential scheduling problems in general.

1.9 Supervised learning tasks

1.9.1 Classification

- **binary**: Assign an example to one of two possible classes (often a positive and a negative one). E.g. digit vs non-digit character.
- **multiclass**: Assign an example to one of $n > 2$ possible classes. E.g. assign a digit character among $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
- **multilabel**: Assign an example to a subset $m \leq n$ of the possible classes. E.g. predict the topics of a text.

1.9.2 Regression

Assign a real value to an example. E.g. predict biodegradation rate of a molecular compound under aerobic conditions.

1.9.3 Ordinal regression or ranking

Order a set of examples according to their relative importance/quality wrt the task. E.g. order emails according to their urgency.

1.10 Unsupervised learning tasks

1.10.1 Dimensionality reduction

Reduce dimensionality of the data maintaining as much information as possible. E.g. principal component analysis (PCA), random projections.

1.10.2 Clustering

Cluster data into homogeneous groups according to their similarity. E.g. cluster genes according to their expression levels.

1.10.3 Novelty detection

Detect novel examples which differ from the distribution of a certain set of data. E.g. recognize anomalous network traffic indicating a possible attack.

1.10.4 Probabilistic Reasoning

- Reasoning in presence of uncertainty.
- Evaluating the effect of a certain piece of evidence on other related variables.
- Estimate probabilities and relations between variables from a set of observations.

1.10.5 Choice of Learning Algorithms

Information available:

- Full knowledge of probability distributions of data: Bayesian decision theory
- Form of probabilities known, parameters unknown: Parameter estimation from training data
- Form of probabilities unknown, training examples available: discriminative methods, do not model input data (generative methods), learn a function predicting the desired output given the input
- Form of probabilities unknown, training examples unavailable (only inputs): unsupervised methods, cluster examples by similarity

2 Decision Trees

A decision tree represents a disjunction of conjunctions of constraints over attribute values. Each path from the root to a leaf is a conjunction of the constraints specified in the nodes along it:

$$OUTLOOK = Overcast \wedge LESSON = Theoretical$$

The leaf contains the label to be assigned to instances reaching it. The disjunction of all paths is the logical formula represented by the tree.

2.1 Appropriate problems

- Binary or multiclass classification tasks (extensions to regressions also exist)
- Instances represented as attribute-value pairs
- Different explanations for the concept are possible (disjunction)
- Some instances have missing attributes
- There is need for an interpretable explanation of the output

2.2 Learning decision trees

- Greedy top-down strategy (ID3 - Quinlan 1986, C4-5 - Quinlan 1993)
- For each node, starting from the root with full training set:
  1. Choose best attribute to be evaluated
  2. Add a child for each attribute value
  3. Split node training set into children according to value of chosen attribute
  4. Stop splitting a node if it contains examples from a single class, or there are no more attributes to test.
- Divide et impera approach

2.3 Chosing the best attribute

2.3.1 Entropy

Entropy is a measure of the amount of information contained in a collection of instances S which can take a number c of possible values:

$$H(S) = -\sum_{i=1}^{c} p_i log_2 p_i$$

where p_i is the fraction of S taking value i.
In our case instances are training examples and values are class labels. The entropy of a set of labelled examples measures its label inhomogeneity.

2.3.2 Information gain

Expected reduction in entropy obtained by partitioning a set S according to the value of a certain attribute A:

$$IG(S, A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v)$$

where $Values(A)$ is the set of possible values taken by A and S_v is the subset of S taking value v at attribute A.
The second term represents the sum of entropies of subsets of examples obtained partitioning over A values, weighted by their respective sizes. An attribute with high information gain tends to produce homogeneous groups in terms of labels, thus favouring their classification.

2.4 Issues in decision tree learning

2.4.1 Overfitting avoidance

Requiring that each leaf has only examples of a certain class can lead to very complex trees. A complex tree can easily overfit the training set, incorporating random regularities not representative of the full distribution, or noise in the data. It is possible to accept impure leaves, assigning them the label of the majority of their training examples.
Two possible strategies to prune a decision tree:

- **pre-pruning** decide whether to stop splitting a node even if it contains training examples with different labels.
- **post-pruning** learn a full tree and successively prune it removing subtrees.

2.4.2 Reduced error pruning

It's a post-pruning strategy. It assumes a separate labelled validation set for the pruning stage.
Procedure:

1. For each node in the tree, evaluate the performance on the validation set when removing the subtree rooted at it
2. If all node removals worsen performance, STOP.
3. Choose the node whose removal has the best performance improvement
4. Replace the subtree rooted at it with a leaf
5. Assign to the leaf the majority label of all examples in the subtree
6. Return to 1

2.4.3 Dealing with continuous-valued attributes

Continuous valued attributes need to be discretized in order to be used in internal nodes tests. Discretization threshold can be chosen in order to maximize the attribute quality criterion (e.g. infogain).
Procedure:

1. Examples are sorted according to their continuous attribute values
2. For each pair of successive examples having different labels, a candidate threshold is placed as the average of the two attribute values.
3. For each candidate threshold, the infogain achieved splitting examples according to it is computed
4. The threshold producing the higher infogain is used to discretize the attribute

2.4.4 Alternative attribute test measures

The information gain criterion tends to prefer attributes with a large number of possible values. As an extreme, the unique ID of each example is an attribute perfectly splitting the data into singletons, but it will be of no use on new examples. A measure of such spread is the entropy of the dataset wrt the attribute value instead of the class value:

$$H_A(S) = - \sum_{v \in Values(a)} \frac{|S_v|}{|S|} log_2 \frac{|S_v|}{|S|}$$

The gain ratio measure downweights the information gain by such attribute value entropy:

$$IGR(S, A) = \frac{IG(S, A}{H_A(S)}$$

2.4.5 Handling attributes with missing values

Assume example x with class c(x) has missing value for attribute A. When attribute A is to be tested at node n, the complex solution implies that at test time, for each candidate class, all fractions of the test example which reached a leaf with that class are summed, and the example is assigned the class with highest overall value.

3 K-Nearest Neighbour

3.1 1-Nearest Neighbour classification

Metric or distance definition: Given a set X, a function $d : X \times X \rightarrow \mathbb{R}_0^+ + 0$ is a metric for X if for any $x, y, z \in X$ the following properties are satisfied:

- reflexivity $d(x, y) = 0 \quad \text{iff} \quad \mathrm{x = y}$
- symmetry $d(x, y) = d(y, x)$
- triangle inequality $d(x, y) + d(y, z) \geq d(x, z)$

3.2 k-Nearest Neighbour classification

```
for all test examples x do
    for all training examples $(x_i , y_i )$ do
        compute distance $d(x, x_i )$
    end for
    select the k-nearest neighbours of x
    return class of x as majority class among neighbours:
```

$$argmax_y \sum_{i=1}^{k} \delta(y, y_i)$$

```
end for
```

Note:

$$delta(x, y) = \begin{cases} 1 & \text{if} \quad \text{x=y} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

3.3 k-Nearest Neighbour regression

```
for all test examples x do
    for all training examples (x i , y i ) do
        compute distance d(x, x i )
    end for
    select the k-nearest neighbours of x
    return the average output value among neighbours:
```

$$\frac{1}{k} \sum_{i=1}^{k} y_i$$

```
end for
```

3.4 Characteristics of k-nearest neighbour learning

- **instance-based learning** the model used for prediction is calibrated for the test example to be processed
- **lazy learning** computation is mostly deferred to the classification phase
- **local learner** assumes prediction should be mainly influenced by nearby instances
- **uniform feature weighting** all features are uniformly weighted in computing distances

3.5 Distance-weighted k-nearest neighbour

Classification:

$$argmax_y \sum_{i=1}^{k} w_i \delta(y, y_i)$$

Regression:

$$\frac{\sum_{i=1}^{k} w_i y_i}{\sum_{i=1}^{k} w_i}$$

where: $w_i = \dfrac{1}{d(x, x_i)}$

4 Linear algebra

4.1 Vector space

A set X is called a vector space over $\mathbb{R}$ if addition and scalar multiplication are defined and satisfy for all $x, y, z \in$ and μ, $\sigma \in \mathbb{R}$:

- Addition:
  - **associative** $x + (y + z) = (x + y) + z$
  - **commutative** $x + y = y + x$
  - **identity element** $\exists 0 \in X : x + 0 = x$
  - **inverse element** $\forall x \in X \; \exists x' \in X : x + x' = 0$
- Scalar multiplication:
  - **distributive over elements** $\lambda(x + y) = \lambda x + \lambda y$
  - **distributive over scalars** $(\lambda + \mu)x = \lambda x + \mu x$
  - **associative over scalars** $\lambda(\mu x) = (\lambda\mu)x$
  - **identity element** $\exists 1 \in \mathbb{R} : 1x = x$

4.1.1 Properties and operations in vector spaces

- **subspace**: any non-empty subset of X being itself a vector space (E.g. projection)
- **linear combination**: given $\lambda_i \in \mathbb{R}$, $x_i \in X$

$$\sum_{i=1}^{n} \lambda_i x_i$$

- **span**: the span of vectors $x_1, \ldots, x_n$ is defined as the set of their linear combinations

$$\left\{ \sum_{i=1}^{n} \lambda_i x_i, \quad \lambda_i \in \mathbb{R} \right\}$$

4.1.2 Basis in vector spaces

- **Linear independency** A set of vectors x i is linearly independent if none of them can be written as a linear combination of the others.
- **Basis** A set of vectors x_i is a basis for X if any element in X can be uniquely written as a linear combination of vectors x_i. Necessary condition is that vectors x_i are linearly independent. All bases of X have the same number of elements, called the *dimension* of the vector space.

5 Probability theory

6 Evaluation

7 Bayesian decision theory