Dispensa Intro to Machine Learning

Bonmassar Ivan

June 29, 2022
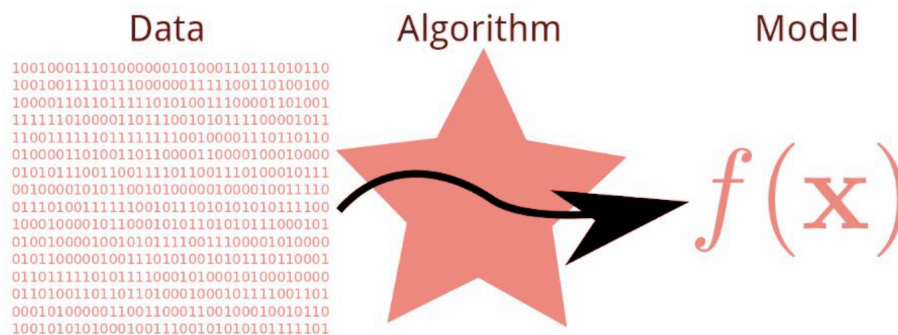
Contents

Chapter 1

ML Basics

The main notion to get from this is the following. ML allows computers to gain **knowledge** acquired through **algorithms** by learning from data. This knowledge is represented through a **model** which is then used on future data.

The training data produces a model or predictor, whereas the testing data produce a prediction.

1.1 Data

Data can be a list of movies from IMDB which is easily representable. Although we need to use **features** when taking into consideration other examples. Features is how an algorithm view data and is generally represented with vectors.

For example, classifying different apples, features could be the shape and colour of them.

The general problem with data for classification for example is that not all data is the same. For instance a banana can either be green or yellow. Although this is true we cannot go to deep with this so we use a probabilistic model called **data generating distribution**. Both training and test data are based on this.

So in our previous example, we will generalize and say that bananas are yellow.

Definition 1.1.1 (Probability distribution)**.** Describes how likely certain events are.

High probability: round apples
Low probability: square apples

1.2   Types of Learning

SUPERVISED LEARNING
Supervised learning is when the algorithm is given labeled examples and the predictor should output a label. A further example of this is **classification**, where the model classifies from a pool of categories.

Given a training set $T = (x_i, y_i)$ learn a function that predicts y given x. x is multi-dimensional.

Some real world examples can be facial recognition, spam detection and character recognition.

Regression is similar to classification only with real values (i.e. numbers).

Ranking the label is a ranking (most similar, most popular web pages etc.)

UNSUPERVISED LEARNING The given data is without labels. Some examples are:

Clustering, where the output is the general structure of the data set (clusters of data). Real world examples are image segmentation, social network analysis

Anomaly detection

Dimensionality reduction

REINFORCEMENT LEARNING

The idea is that the agent interacts with the environment and receives rewards based on behavior.

1.3   ML Ingredients

TASK

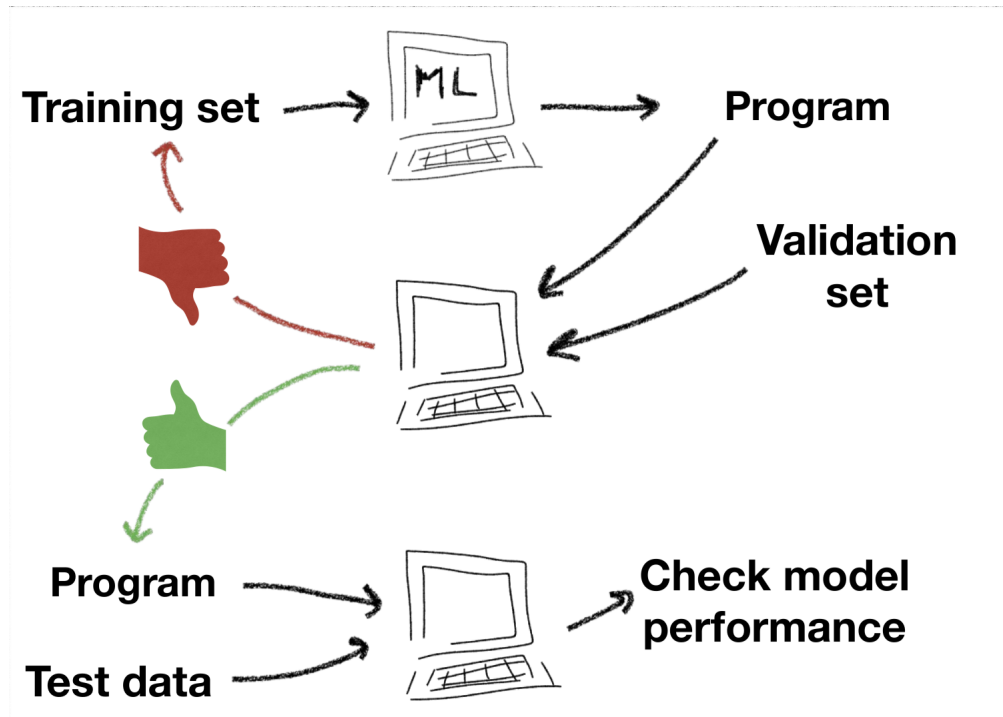A task represents the type of prediction being made to solve a problem.

Assigning each input $x \in \mathscr{X}$ to an output $y \in \mathscr{Y}$

Data

Data is basically the information required to solve a specific problem and as said previously is usually sampled from an unknown data generating distribution :
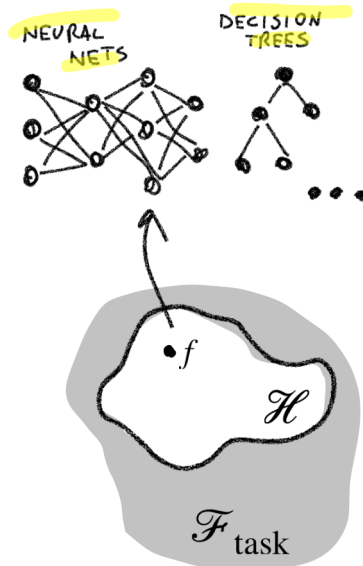
$\mathbf{p}_{data}$

For classification and regression $\mathbf{p}_{data} \in \triangle(\mathscr{X} \times \mathscr{Y})$

Model and hypothesis space

A model is like a program that solves the problem. There are various models (decision trees, neural networks..) and a set of them makes up the hypothesis space.

The objective

The objective is to minimize an error function $E(f, \mathbf{p}_{data})$ to find the optimal function

$f^\star = \arg\min E(f, \mathbf{p}_{data})$.

This however is really hard to do because of the too large search space.

The *feasible target* is the optimal one in a restricted hypothesis space $\mathscr{H}$

$f^\star_{\mathscr{H}} = \arg\min E(f, \mathbf{p}_{data})$.

This is also not doable because we do not have access to $\mathbf{p}_{data}$

The *actual target* is then the following:

$f^\star_{\mathscr{H}}(\mathscr{D}_n) = \arg\min E(f, \mathscr{D}_n)$. where $\mathscr{D}_n$ is a training set.

The error function is usually specified as a pointwise-loss (f, z) measuring the error of f on the training set z.

The learning algorithm solves the optimization problem which targets the actual target.

Chapter 2

K-Nearest Neighbor

Imagine assigning each feature a numeric value, and putting them as coordinates in a feature space. To find which label to give to an example d we can see the k-nearest neighbors. The correct label will be the one that has the majority in k neighbors.

In cases where features are comparable (each feature has the same units) we will use the simple method of Euclidean Distance:

$$D(a, b) = \sqrt{(a_1 - b_1)^2 + (a_n - b_n)^2}$$

2.1 Decision boundaries

Definition 2.1.1 (Decision boundaries). Decision boundaries are places where the the classification of an example changes.

These boundaries are a subset of the Voronoi diagram which divides the initial points distribution by equidistant lines.

(a) Initial points distribution   **(b)** Voronoi diagram by initial points
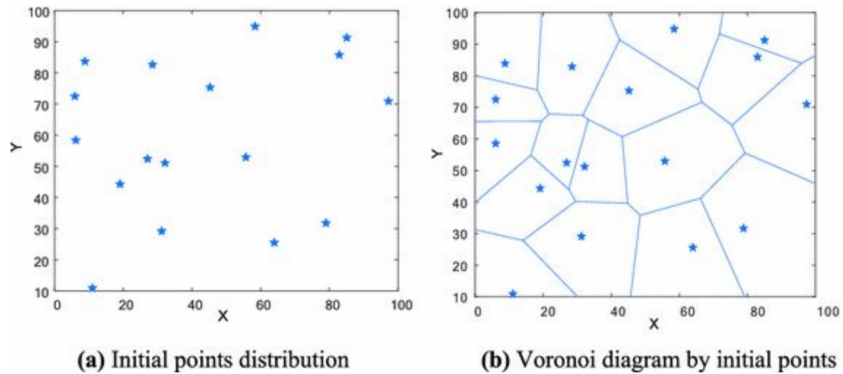
Choosing K will have an effect on underfitting and overfitting.

Definition 2.1.2 (Underfitting)**.** Underfitting occurs when the model is not able to achieve a low error value on the training set

Definition 2.1.3 (Overfitting)**.** Overfitting occurs when the gap between training set error and test set error is large

Some heuristics come into play. It's important to choose an odd number to avoid ties. A general rule of thumb is that k should be lower than the square root of N training examples.
Weighted k-nn, uses the same mechanics only that the examples are weighted which means that they will weigh in more in a vote situation.

Lazy learning is when an algorithm simply stores data and operates when given a test example.

Eager learning on the other hand is when given a training set, the algorithm constructs a classification model which then reutilizes.

2.2 Summary

When is it useful:

- Few features per instance

- Lots of training data

Advantages:

- Training is very fast (lazy)

- Learn complex functions

Disadvantages:

- Slow at query time

The main issue has to do with dimensionality. Every algorithm requires a dense data set, although K-NN requires to have at least one neighbor in each dimension, which makes for a very heavy computational load.

Chapter 3

Linear Models

3.1 Model assumptions

Assumptions (when correct) are a great tool to improve an algorithm. They can however lead to overfitting. For K-NN the only assumption that's made is the fact that proximity relates to label.

Definition 3.1.1 (Bias)**.** The bias of a model is how strong its assumptions are.

K-NN and Decision Trees are considered to be low bias.

3.2 Linear models

A high-bias assumption is linear separability, which means that the classes can be simply divided by a line (or hyperplane in n dimensions).

A line is defined by a pair of values:

$$0 = w_1 f_1 + w_2 f_2$$

To classify we can input the values into the equation. If it's positive it will be above the line and viceversa.

In n-dimensions:

$$0 = b + \sum_{i=1}^{n} w_i f_i$$

3.3 Online learning

This linear model is different because it sees one example at a time. This is useful when taking in data streams (for example from online resources).

The algorithm receives an unlabeled example, it predicts and if it is not correct it updates its model.

3.4 Perceptron algorithm

This is a linear model that constantly updates when not correct.

repeat until convergence (or for some # of iterations):

for each training example (f_1, f_2, ..., f_n, label):

$$prediction = b + \sum_{i=1}^{n} w_i f_i$$

if not correct, update all the weights:

for each w_i:

$w_i = w_i + f_i$*label

$b = b + $ label

Chapter 4

Multiclass Classification

Definition 4.0.1 (Binary classification). Given an input $\mathscr{X}$, an unknown distribution D over $\mathscr{X} \times \{-1, 1\}$ a training set D compute a function f.

Definition 4.0.2 (Multiclass classification). Given an input $\mathscr{X}$ and a number K of classes, an unknown distribution D over $\mathscr{X} \times K$ a training set D compute a function f.
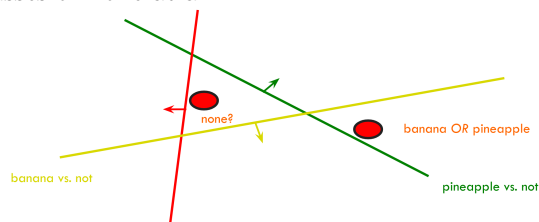
None of the current methods work.

The main idea is to modify the perceptron by adding multiple lines. There are two approaches to this.

4.1 One vs All (OVA)

For each label L define a binary problem. All examples with L are positive, the rest is negative.

This poses a problem related to some areas in which it can either be two classes or none at all.

How to classify?

Generally speaking the classifier should provide a level of confidence. To calculate this value we need to use decision boundaries and its distance from the hyperplane.

4.2 All vs All (AVA)

The idea behind AVA is to pair up each permutation of 1v1 pairs. Then, the classifier will receives all examples of i as a +1 and -1 for the j class. When a test point arrives we evaluate on all the F_{ij} classifiers.

4.3 Summary

AVA has faster training time but a slower test time. AVA has more chances of error at test time.

Evaluation comes when we need to evaluate a specific performance of an algorithm. Can be done through microaveraging, average over the examples, or macroaveraging, average of the average of each label.

Chapter 5

Gradient Descent

5.1 Model based machine learning

There are 3 steps to this model:

1. Pick a model (DT, perceptron)

2. Pick a criterion to optimize

3. Develop a learning algorithm

In linear model:

1. $0 = $ b$+ \sum_{j=1}^{n} w_j f_j$

2. $\sum_{i=1}^{n} 1[y_y(w \cdot x_i + b) \leq 0]$ (0/1 Loss function aka the total number of mistakes)

3. $argmin_{w,b} \sum_{i=1}^{n} 1[y_y(w \cdot x_i + b) \leq 0]$

5.2 Surrogate loss functions

Minimizing the 0/1 Loss function is an NP-hard problem, because of the many local minima and any change in w can change drastically the result. Ideally we would want a convex function so to have at least one minimum.

By **surrogate loss function** we mean a loss function that provides an upper bound to the 0/1 loss function.

- 0/1 loss : $l(y, y') = 1[yy' \leq 0]$

- Hinge loss: $l(y, y') = max(0, 1 - yy')$

- Exponential loss: $l(y, y') = exp(yy')$

- Squared loss: $l(y, y') = (y - y')^2$

Surrogate loss functions
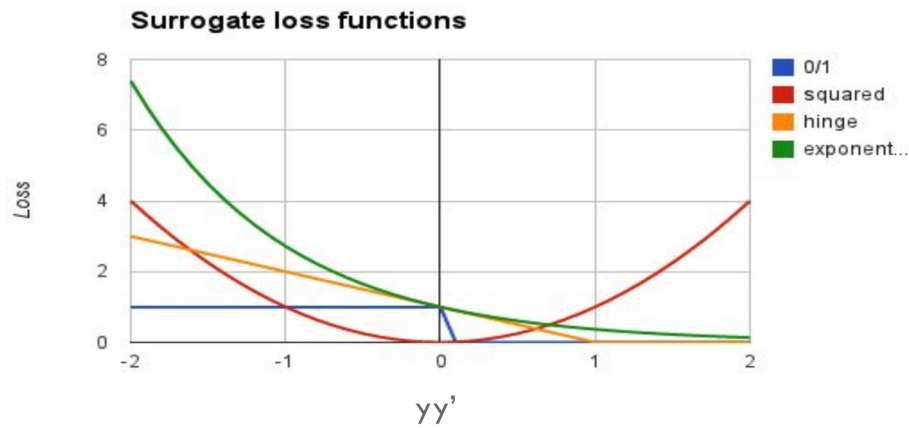
5.3   Gradient descent

We use gradient descent to find a minimum in the function.

The general approach is to choose a starting point w, calculate the derivative and move in that direction.

$$w_j = w_j - \eta \frac{d}{dw_i} loss(w)$$

mi piace il pene

η is the rate at which we move and will change over time.

Since it's changing over time it can be implemented through a perceptron algorithm.

The main problem with gradient descent is local minima: