Algorithmic Foundations of Datascience

A summary of the Lecture

Felix F.

Summer term 2019

Abstract

This summary is based on the lecture 'Algorithmic Foundations of Datascience' held by Prof. Martin Grohe in the summer term of 2019 at RWTH Aachen University.

Since this summary is made by student(s), there are definitely mistakes in the script. Should you find one feel free to contact me or push a correction to the repository.

Contents

1 Machine Learning Basics

1.1 Definitions

All the following definitions are specific to 'Machine Learning' and might change throughout the summary.

Data is viewed as a collection of data items (or dataset). A **data item** is represented by a so-called feature vector. The **feature vector** basically a list of features and attributes in form of a vector

1.1.1 Domains

The **Domain** $\mathbb{D}$ is a set containing the range of all possible values for a single feature. The **Instance Space** $\mathbb{X}$ is the Cartesian product of the domains of all features in the feature vector. The number of features is the dimension of the instance space (or the number of entries in the feature vector).

◀ $\mathbb{D}$
◀ $\mathbb{X}$

> *Example:* The domain space can be something like $\{true, false\}$ or $\mathbb{N}$, aso. The instance space of the domain spaces is then (informally) $\mathbb{D}_1 \times \mathbb{D}_2 \times \ldots \times \mathbb{D}_l$

Assuming n is the number of data items, and l the number of features the whole dataset can be viewed as an $(n \times l)$-matrix over the instance space.

1.2 Forms of Learning

Unsupervised learning tries to detect patterns in data with, hence the name, no feedback, supplied the Learning algorithm. This technique is mostly used for clustering.

Reinforcement Learning tries to find actions that maximise reward. The feedback to the learning algorithm is provided in form of a reward (or punishment)

In **Supervised learning** one tries to approximate a function by training on input-output pairs. **Classification** has a finite-valued function and tries to predict values for future inputs. **Regression** has a numerical function and tries to predict the expected values for future inputs. Supervised learning can be differentiated into two different set-ups. First, **Batch Learning**, in which all examples are supplied at once and the learner has to come up with a hypothesis. Where as in **Online Learning**, examples are given 'on-the-fly' once at a time and the initial hypothesis gets improved over time.

Semi-supervised Learning is similar to supervised learning, but takes only a few and possibly faulty examples and tries to make the best of the possibly small and faulty dataset.

1.3 Hypothesis Space

$\mathcal{H}$ ▶ In Supervised learning, the unknown target function h is choosen from the **hypothesis space** $\mathcal{H}$, hence $h \in \mathcal{H}$. The Hypothesis space contains all linear functions, all polynomials or all polynomials of a pre-scribed degree, and all functions that can be described by a decision tree.

A learning problem is **Realisable** if the target function h is in the Hypothesis Space.

> *Example:* Assuming a target function $h(x) := a \cdot \sin(x) + bx + c$. Then, if $\mathcal{H}$ only contains polynomials, the learning problem is **not realizable**. But, assuming $\mathcal{H}$ contains linear functions or polynomials in x and $\sin(x)$, the learning problem **is realisable**.

1.4 Validation

The goal of the learning algorithm is to produce a hypothesis that generalizes well, that is, approximates the target function well ion the all data points (and not only those in the training set). To evaluate how well a hypothesis generalizes we can evaluate it against some test set. A **test set** is generates by splitting the examples into a test set and training set.

The empirical observation that simpler hypothesis tend to generalise better is picked up by **Occam's Razor** which states, that the simplest hypothesis which is consistent with the data should be choosen.

Overfitting is the phenomenon, which occurs when the leaner tries to match the training examples as exactly as possible which leads to complex hypotheses that generalize badly. Thus, to avoid overfitting, it is often better to choose a simple hypothesis even if it doesn't match the data exactly.

1.5 Nearest Neighbour Learning

The underlying idea of this simple learning algorithm is to predict the value of a function at point x by looking at known values of points close to the given one and assume the value of x is similar.

1.5.1 Metric

d ▶ A **metric** d on $\mathbb{X}$ is a function $d : \mathbb{X}^2 \longrightarrow \mathbb{R}$ such that for all $x, y, z \in \mathbb{X}$

Nonnegativity	**Symmetry**	**Triangle Inequality**
$d(x,y) \geq 0$ and $d(x,y) = 0 \iff x = y$;	$d(x,y) = d(y,x)$	$d(x,z) \leq d(x,y) + d(y,z)$

1.5.2 Metric Spaces

Suppose $\mathbb{X}$ is the instance space, and d be some metric on $\mathbb{X}$. Then we have a **Metric Space** $(\mathbb{X}, d)$. ◀ $(\mathbb{X}, d)$

1.5.2.1 Common distance metrics

The distance between points x and y is denoted as $d(x, y)$. ◀ $d(x, y)$

Euclidean distance with $\mathbb{X} = \mathbb{R}^\ell$:

$$d(x, y) = \sqrt{\sum_{i=1}^{\ell} (x_i - y_i)^2} \tag{1.5.1}$$

Manhattan distance with $\mathbb{X} = \mathbb{R}^\ell$:

$$d(x, y) = \sum_{i=1}^{\ell} |x_i - y_i| \tag{1.5.2}$$

Hamming distance with $\mathbb{X} = \mathbb{D}_1 \times \mathbb{D}_2 \times \ldots \times \mathbb{D}_\ell$

$$d(x, y) = |\{\, i \mid x_i \neq y_i \}| \tag{1.5.3}$$

1.5.3 Nearest Neighbour Classifier

Problem description: 'Learn' a function f, that associates a class $f(x) \in \mathbb{Y}$ with every $x \in \mathbb{X}$. Basically: $f : \mathbb{X} \to \mathbb{Y}, x \mapsto y$. Function f is provided with a dataset to learn from: $(x_1, y_1), \ldots, (x_m, y_m)$ with m entries where each entry corresponds to $(x_i, f(x_i))$. x_i is called data item and y_i is called class x_i.

k-Nearest Neighbour Classifier uses $x \in \mathbb{X}$ as input.

1. Find the k nearest neighbours $x_{i_1}, \ldots, x_{i_k}$ of x in $\{x_1, \ldots, x_m\}$
2. Take a 'majority vote' of the class that appears the most among $y_{i_1}, \ldots, y_{i_k}$

Choosing an appropriate parameter k: When choosing $k = 1$ the classifier tends to overhit, while with larger k one might start to oversimplify. Obviously, the optimal value for k is difficult to determine and depends heavily on the application.

1.5.4 Learning Decision Trees

A decision tree is a rooted tree with labelled notes and edges

- Every internal node of the tree is labelled by an (input) feature
- Every edge is labeled by a value or range of values for the feature labelling its source node.
- Every leaf is labeled with an output value

Semantics: The function value for an input vector x is the value at the leaf of the unique path in the tree whose edges are labelled by the feature values in x.

Note: Computing a smallest decision tree for a given sets of examples is NP-Complete

1.5.4.1 Greedy Algorithm for Building Decision Trees

TODO: Greedy Algorithm for Building Decision Trees

1.6 The Perceptron

The Perceptron algorithm attempts to find a linear classifier for a Boolean classification problem.

1.6.1 Scalar product

$\langle \mathbf{x}, \mathbf{y} \rangle$ ▶ Two vectors, $\mathbf{x} = (x_1, \ldots, x_l)^T$, $\mathbf{y} = (y_1, \ldots, y_l)^T \in \mathbb{R}^\ell$

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{\ell} x_i y_i$$

1.6.2 Euclidean norm

$\|\mathbf{x}\|$ ▶ Vector $\mathbf{x} \in \mathbb{R}^\ell$

$$\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle} = \sqrt{\sum_{i=1}^{\ell} x_i^2}$$

1.6.3 Cauchy-Schwarz Inequality

$$|\langle \mathbf{x}, \mathbf{y} \rangle| \leq \|\mathbf{x}\| \cdot \|\mathbf{y}\| \tag{1.6.1}$$

1.6.4 Hyperplanes

A (affine) **Hyperplane** in $\mathbb{R}^\ell$ is an affine subspace of dimension $\ell - 1$. It can be described as the set of all solutions $\mathbf{x} \in \mathbb{R}^\ell$ to the equation $a_1 x_1 + \ldots + a_\ell x_\ell = b$ where $a_1, \ldots, a_\ell, b \in \mathbb{R}$. In short: $\langle \mathbf{a}, \mathbf{x} \rangle - b = 0$, or more formally: $P = \{x \in \mathbb{R}^\ell \mid \langle \mathbf{a}, \mathbf{x} \rangle = b\}$.

A Hyperplane $P = \{x \in \mathbb{R}^\ell \mid \langle \mathbf{a}, \mathbf{x} \rangle = b\}$ is **homogeneous** if $b = 0$, or equivalently, if $0 \in P$. That's, the Hyperplane goes through the origin.

1.6.5 Halfspaces

A **Halfspace** in $\mathbb{R}^\ell$ is the set of all points on one side of a Hyperplane. A Halfspace can be described as the set of solutions $\mathbf{x} \in \mathbb{R}^\ell$ to an inequality $\langle \mathbf{a}, \mathbf{x} \rangle - b \geq 0$ where $\mathbf{a} \in \mathbb{R}^\ell$ and $b \in \mathbb{R}$. Or more formally: $H = \{x \in \mathbb{R}^\ell \mid \langle \mathbf{a}, \mathbf{x} \rangle - b \geq 0\}$

A Halfspace is a **homogeneous Halfspace** as follows: $H = \{x \in \mathbb{R}^\ell \mid \langle \mathbf{a}, \mathbf{x} \rangle \geq 0\}$. So basically a halfspace with $b = 0$.

1.6.6 Linear Classification

The **Boolean Classification** is a classification problem with the instance space $\mathbb{X} = \{+1, -1\}$. The goal is to learn an unknown target function $f : \mathbb{R}^\ell \to \{+1, -1\}$.

The input for the learning function is a sequence of data items (so called training sequence):

$$S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)) \in \mathbb{R}^\ell \times \{+1, -1\} \tag{1.6.2}$$

The hypothesis space ($\mathcal{H}$) consists of **linear separators**, which are functions $h : \mathbb{R}^\ell \to \mathbb{R}$ in the form of:

$$h(\mathbf{x}) = \operatorname{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle - b) = \begin{cases} +1 & \text{if } \langle \mathbf{w}, \mathbf{x} \rangle - b > 0 \\ 0 & \text{if } \langle \mathbf{w}, \mathbf{x} \rangle - b = 0 \\ -1 & \text{if } \langle \mathbf{w}, \mathbf{x} \rangle - b < 0 \end{cases} \tag{1.6.3}$$

with $w \in \mathbb{R}^\ell$, the weight vector and $b \in \mathbb{R}^\ell$ the bias.

Because of the mismatching target function range of $\{+1, 0, -1\}$ and the instance space of the Boolean Classification $\{+1, -1\}$ the linear separator function is redefined as follows:

$$f(\mathbf{x}) = \begin{cases} +1 & \text{if } \langle \mathbf{w}, \mathbf{x} \rangle - b \geq 0 \\ -1 & \text{if } \langle \mathbf{w}, \mathbf{x} \rangle - b < 0 \end{cases} \tag{1.6.4}$$

A hypothesis h is an **consistent hypothesis** if for a training sequence $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$ the following applies $h(\mathbf{x}_i) = y_i$ for all $i \in [m]$.

1.6.6.1 Non-bias Linear Classification

A **homogeneous linear separator** is a function, such that $\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle$. The removal of the bias (b) implies, the linear separator goes through the origin of the coordinate system.

THEOREM Properties of non-bias Linear Classification (1.6.5)

Let $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$ with $\mathbf{x}_i = (x_{i_1}, \ldots, x_{i_\ell}) \in \mathbb{R}^\ell$ and $y_i \in \{-1, 1\}$ and $\mathbf{w} = (w_1, \ldots, w_\ell) \in \mathbb{R}^\ell$ and $b \in \mathbb{R}$. Then the following are equivalent.

1. $\mathbf{x} \mapsto \operatorname{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle - b)$ is a linear separator consistent with S

2. $\mathbf{x}' \mapsto \text{sgn}(\langle \mathbf{w}', \mathbf{x}' \rangle)$ is a homogeneous linear separator consistent with $S' = ((\mathbf{x}'_1, y_1), \ldots, (\mathbf{x}'_m, y_m))$, where $\mathbf{x}'_i := (x_{i_1}, \ldots, x_{i_\ell}, 1)$ and $\mathbf{w}' = (w_1, \ldots, w_\ell, -b)$

1.6.6.2 Normalized Linear Classification

Intuition: as the name suggests informally, we simply divide all $\mathbf{x}$ in the training sequence by the length of the largest $\mathbf{x}$ vector. Therefore, the largest vector then has a length of at most 1.

Suppose we have a training sequence $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$ with $\mathbf{x}_i = (x_{i_1}, \ldots, x_{i_\ell}) \in \mathbb{R}^\ell$ and $y_i \in \{-1, 1\}$. We can apply the following transformation to all datapoints $\mathbf{x}_i$

$$\mathbf{x}_i \mapsto \widehat{\mathbf{x}}_\mathbf{i} := \frac{\mathbf{x}'_i}{\max_{1 \leq j \leq m} \|\mathbf{x}'_j\|} \tag{1.6.6}$$

Resulting in the **normalized training sequence** $\widehat{S} = ((\widehat{\mathbf{x}}_1, y_1), \ldots, (\widehat{\mathbf{x}}_m, y_m))$. Additionally $\widehat{S}$ has following properties:

1. $\widehat{S}$ has a homogeneous linear separator if and only if S has a linear separator
2. $0 < \|\widehat{\mathbf{x}}_\mathbf{i}\| \leq 1$ for all $i \in [m]$

1.6.6.3 Linear Seperator Margin

The **margin of a linear seperator** is the orthogonal distance of the nearest point to the linear separator function h. Formally: the margin of h with respect to the target sequence S is

$$\min_{(x,y) \in S} |\langle \mathbf{w}, \mathbf{x} \rangle| \tag{1.6.7}$$

THEOREM (1.6.8)

Let S be a normalized sequence of examples such that there is a homogeneous linear separator consistent with S of margin γ. Then the perceptron algorithm applied to S finds a linear separator after at most $\frac{1}{\gamma^2}$updates of w.

PROOF See Page 1.46 in the slidedeck for Kap1

1.6.6.4 The Perceptron Algorithm

TODO: Add Perceptron Algorithm

1.6.6.5 Linear Separation for Nonlinear Problems

1.7 Neural Networks

1.8 k-Means Clustering

1.8.1 Centroid Based Clustering

1.8.2 k-Means Clustering Algorithm