Machine learning from scratch

Lecture 1: Mathematical background

Alexis Zubiolo
alexis.zubiolo@gmail.com

Data Science Team Lead @ Adcash

January 26, 2017

Before we start

IT STEP will be organizing a Tech night on **February 16th** (Thursday) from 7pm. I will (probably) be giving a talk. The course will most likely be postponed.

Motivation, vocabulary and notations

In **supervised learning** tasks, we are given a *data set* of the form:

$$D = \left\{ \left(\boldsymbol{x}^{(i)}, y^{(i)} \right), \boldsymbol{x}^{(i)} \in \mathcal{X}, y^{(i)} \in \mathcal{Y}, i \in \{1, \ldots, n\} \right\}$$

Motivation, vocabulary and notations

In **supervised learning** tasks, we are given a *data set* of the form:

$$D = \left\{ \left(\boldsymbol{x}^{(i)}, y^{(i)} \right), \boldsymbol{x}^{(i)} \in \mathcal{X}, y^{(i)} \in \mathcal{Y}, i \in \{1, \dots, n\} \right\}$$

- n is the size of the data set (number of *instances*/*samples*)
- In most applications:
  - $\mathcal{X} = \mathbb{R}^d$ (d is the *dimensionality*)
  - $\mathcal{Y} = \mathbb{R}$ (*regression*) or $\mathcal{Y} \subset \mathbb{N}$ (*classification*)
- $\boldsymbol{x} \in \mathcal{X}$ is the *feature vector* and $y \in \mathcal{Y}$ is the *label*

Motivation, vocabulary and notations

Solving a **supervised learning problem** is finding (or *learning*) a function (or *hypothesis*) $h : \mathcal{X} \mapsto \mathcal{Y}$ such that for $(\boldsymbol{x}, y) \in D$, $h(\boldsymbol{x})$ is a *good* estimation (or approximation) of y.

Motivation, vocabulary and notations

Solving a **supervised learning problem** is finding (or *learning*) a function (or *hypothesis*) $h : \mathcal{X} \mapsto \mathcal{Y}$ such that for $(\boldsymbol{x}, y) \in D$, $h(\boldsymbol{x})$ is a *good* estimation (or approximation) of y.

We often write $h(\boldsymbol{x}) = \hat{y}$ ($\hat{y}$ is the *prediction* of $\boldsymbol{x}$ by h).

Motivation, vocabulary and notations

Solving a **supervised learning problem** is finding (or *learning*) a function (or *hypothesis*) $h : \mathcal{X} \mapsto \mathcal{Y}$ such that for $(\boldsymbol{x}, y) \in D$, $h(\boldsymbol{x})$ is a *good* estimation (or approximation) of y.

We often write $h(\boldsymbol{x}) = \hat{y}$ ($\hat{y}$ is the *prediction* of $\boldsymbol{x}$ by h).

This raises **2 questions**:

- How to define h?
- How to assess whether $\hat{y}$ is a good approximation of y?

Hypothesis parametrization

h is often defined by a parameter vector θ and can be noted h_θ.

Hypothesis parametrization

h is often defined by a parameter vector θ and can be noted h_θ.
Several ways to parametrize h exist:

- **Linear model**: $h(\boldsymbol{x}) = \theta^T \boldsymbol{x}$
- **Polynomial kernel** (degree k): $h(\boldsymbol{x}) = \left(1 + \theta^T \boldsymbol{x}\right)^k$
- Other kernels exist, more on this when we talk about duality
- With a **neural net**, more on this later as well
- ...

Hypothesis parametrization

h is often defined by a parameter vector θ and can be noted h_θ.
Several ways to parametrize h exist:

- **Linear model**: $h(\boldsymbol{x}) = \theta^T \boldsymbol{x}$
- **Polynomial kernel** (degree k): $h(\boldsymbol{x}) = \left(1 + \theta^T \boldsymbol{x}\right)^k$
- Other kernels exist, more on this when we talk about duality
- With a **neural net**, more on this later as well
- ...

How you define h highly depends on the application, for example:

- Sometimes a lot of data preprocessing has been made and a simple model (e.g. linear) would work well
- You might have **time/hardware constraints**: In this case going for a too complex model might be crippling
- For neural net, the architecture depends a lot on the type of data you have

Loss functions

Recall we want to know whether y a good prediction of $\hat{y}$.

Regularization

Optimization

Now, we want to minimize a function of the form

$$J(\theta) = \sum_{i=1}^{n} \ell\left(y^{(i)}, \hat{y}^{(i)}\right) \tag{1}$$