MaMa Summary

October 20, 2023

Contents

1 Classification Learning **2**

1.1 vocabulary 2

1.2 loss and error 2

1.3 logistic regression 3

1.4 training error and real life 3

1.5 Quadratic classifier 4

1.6 nearest neighbour classifier 4

1.7 decision tree classifier 4

1.8 loss functions 5

1 Classification Learning

1.1 vocabulary

- domain set X: set of possible inputs
- classes or label set Y: target of classification
- data points or samples: $x \in X$
- features or attributes: entries of x
- training set $S \subseteq X$: pairs (x, y) of data points x and labels y
- classifier h: function $X \to Y$

1.2 loss and error

Let h be a classifier $h : X \to Y$.

- loss function: $l(y, y') \geq 0$ for $y, y' \in Y$.
- zero-one loss:
$$l_{0-1}(y, y') = \begin{cases} 0, & \text{if } y = y' \\ 1, & \text{if } y \neq y' \end{cases}$$
- training error:
$$L_S(h) = \frac{1}{|S|} \sum_{(x,y) \in S} l(y, h(x))$$

Example 1.2.1. *Let $X = \mathbb{R}^2$ and $Y = \{-1, 1\}$. The easiest function is a linear classifier that splits X in two parts and characterizes data points accordingly. For the general case of $X = \mathbb{R}^n$, a $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$ we have:*

$$h_{w,b} = sgn(w^T x + b)$$

You can get rid of the bias b by adapting the training set a little:

$$\overline{S} = (\begin{pmatrix} x \\ 1 \end{pmatrix}, v) : (x, y) \in S$$

where the new linear classifier is defined by

$$\overline{h}_{\overline{w}}(x) = sgn(\overline{w}^T \overline{x})$$

with $\overline{w} = \binom{w}{b}$.

When is it possible to achieve training error 0? (wrt zero-one-loss)
This is precisely then the case, when $\exists w \in \mathbb{R}^n$ s.t. $\forall (x, y) \in S$

- if $y = 1$ then $w^T x \geq 0 \leftrightarrow$ if $y = 1$ then $w^T x > 0$
- if $y = -1$ then $w^T x < 0$

Both can be generalized by saying

$$y \cdot w^T x > 0$$

or

$$\exists w : y \cdot w^T x \geq 1 \; \forall (x, y) \in S$$

A training set with 0 training error is called separable.

1.3 logistic regression

Logistic regression computes a linear classifier.
direct way:
Look for a $w \in \mathbb{R}^n$ such that the training error is minimised, i.e.

$$\min_{w \in \mathbb{R}^n} \frac{1}{|S|} \sum_{(x,y) \in S} h_{0-1}(y, h_w(x))$$

Definition 1.3.1 (logistic function)**.** The logistic function $\phi_{sig} : \mathbb{R} \to [0, 1]$ is defined by

$$z \mapsto \frac{1}{1 + e^{-z}}$$

We then solve

$$\min_{w \in \mathbb{R}^n} \frac{1}{|S|} \sum_{(x,y) \in S} -\log_2(\phi_{sig}(y w^T x))$$

Lemma 1.3.2. *For all training sets* $S \subseteq \mathbb{R}^n \times \{-1, 1\}$ *it holds that*

$$\frac{1}{|S|} \sum_{(x,y) \in S} h_{0-1}(y, h_w(x)) \leq \frac{1}{|S|} \sum_{(x,y) \in S} -log_2(\phi_{sig}(y w^T x))$$

1.4 training error and real life

A classifier is only good, if it performs good on new data. To test your classifier you need to evaluate the classifier on data it hasn't seen during training. This is called the

test error. The question now is, how the data should be split into test and training data. This is a difficult question and should be evaluated for each use case separately. A part from the training set should also be used for validation if there are degrees of freedom in the chosen classifier.

1.5 Quadratic classifier

This is another binary classifier. In this case wa search for a matrix $U \in \mathbb{R}^{n \times n}$ of weights, a vector $w \in \mathbb{R}^n$ and a bias $b \in \mathbb{R}$. The classifier is then defined by

$$h(x) = sgn\left(\sum_{i,j=1}^{n} u_{i,j} x_i x_j + \sum_{i=1}^{n} w_i x_i + b\right) = sgn(x^T u x + w^T x + b)$$

To simplify things we redefine the training set S by $\overline{S}$ containing all samples (x, y) with $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ and y analogously defined. Redefine x by

$$\overline{x} = (x_1^2, x_1 x_2, x_1 x_3, .., x_n^2, .., x_n, 1)$$

Doing this will transform the quadratic classifier into a linear one.

1.6 nearest neighbour classifier

The training set again is defined by $S \subseteq X \times Y$. When adding a new data-point, we check what class the nearest data point belongs to and add the new point to the same class. In the case of k-nearest-neighbour we check the classes of the k nearest points. The problem is that the training set must be in memory the entire time. This makes the learning potentially very slow.

1.7 decision tree classifier

The classifier of a decision tree works as follows:

1. Set $v = r$ where r is the tree's root
2. while v is not a leaf do
3. Let (i, j) be the decision rule of v
4. If $x_i \leq t$ set $v = v_L$ else set $v = v_R$
5. end while

6. Output the class $c(v)$ of v

The following now describes how a tree can be learned.
The idea is to start with a single node r where the class $c(r)$ is just the majority class in S.
Iteratively decide for each leaf v, if it is beneficial to split it into two child nodes. Let S_v be the path of the training set that, following the existing decisions, ends up in v. The split into S_L and S_R would then be defined by a feature i and a threshold t. We define the gain of the split by

$$gain(v, i, t) = \gamma(S_v) - \left(\frac{|S_L|}{|S_R|}\gamma(S_L) + \frac{|S_R|}{|S_L|}\gamma(S_R) \right)$$

where γ is a inhomogeneity measure. This can be defined in multiple ways.

Definition 1.7.1. The inhomogeneity can be defined by the training error

$$\gamma(S_v) = 1 - \max_{y \in Y} p(y, S_v)$$

where

$$p(y, S_v) = \frac{|\{(x, y') \in S_v : \ y' = y\}|}{|S_v|}$$

Definition 1.7.2 (gini impurity)**.** This is used in scikit-learn.

$$\gamma(S_v) = 1 - \sum_{y \in Y} p(y, S_v)^2$$

Definition 1.7.3 (entropy)**.**

$$\gamma(S_v) = -\sum_{y \in Y} p(y, S_v) \cdot \log_2(y, S_v)$$

1.8 loss functions

So far, we only looked at the zero-one-loss $l_{0-1}(y, y') = \begin{cases} 1, & y \neq y' \\ 0, & y = y' \end{cases}$ where y is the true class and y' is the prediction.
Consider the use case of a spam filter. This is a binary classifier that checks whether a new mail is spam or not. This admits two kind of errors:

- false-positive: good mail is classified as spam
- false-negative: spam mail is classified as good

The case of false-positive is more serious in this example. This needs to be applied to the loss function.

$$l(y, y') = \begin{cases} 0, \ y = y' \\ 10, \ \text{if } y \text{ is good and } y' \text{ spam} \\ 1, \ \text{if } y \text{ is spam and } y' \text{ is good} \end{cases}$$

Example 1.8.1 (loss functions in regressions)**.** *Consider a predictor* $h : \mathbb{R}^n \to \mathbb{R}$*. The (mean)square-loss is defined by*

$$l(y, y') = (y - y')^2$$

and the (mean) absolute loss

$$l(y, y') = |y - y'|$$