Introduction to Machine Learning

Perceptron

Guillaume Wisniewski
guillaume.wisniewski@limsi.fr
February 2017
Université Paris Sud — LIMSI

Supervised Classification

Our goal...

Classification

- $\mathcal{X} \rightarrow$ set of all examples/observations (in practice $\mathcal{X} = \mathbb{R}^n$)
- $\mathcal{Y} \rightarrow$ set of labels
- there is functional mapping between an observation and a label : given an observation, an expert (you ?) can find the label
- finding the label of an observation = classify the observation
- $\mathcal{Y}$ = discrete space (classification) or $\mathcal{Y} = \mathbb{R}^n$ (regression)

Formally, a classifier is :

$$\begin{aligned} f : \mathcal{X} &\mapsto \mathcal{Y} \\ x &\rightarrow y \end{aligned}$$

Examples

Observations

- features : words, number of smiley, time-stamps, language, ...

$\Rightarrow$ binary classification

Label

1. spam
2. not spam

Example (2)

Observation

- features : number of white pixel, number of red pixels, number of persons, ...

Labels

- basket
- football
- field hockey
- other

$\Rightarrow$ multi-class classification

Example (3)

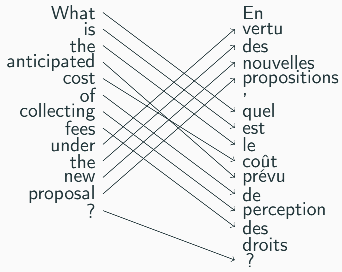
He reckons the current account deficit will narrow to only $1.8 billion in September.

$\downarrow$

[$_{NP}$He] [$_{VP}$reckons] [$_{NP}$the current account deficit] [$_{VP}$will narrow] [$_{PP}$to] [$_{NP}$only $1.8 billion] [$_{PP}$in] [$_{NP}$September.]

Sequence labeling ($\simeq$ multi-class classification with a large set of labels = structure prediction)

Example (4)



Word alignment (structure prediction)

Example (5)

Comment va ton frère ?

↓

τι κανει ο αδελφος σου ;

Example (5)

Comment va ton frère ?

↓

τι κανει ο αδελφος σου ;

as many classes as there are possible sentences in Greek

Formalization

Supervised Learning

- f is not known
- f can not be formally defined : fuzzy decision $\oplus$ many criteria must be considered
- goal : learn/estimate/infer f from a set of labeled data
- supervised learning : we know the label of all training examples

Problematic

- Find a function that can predict the label of 'most' examples from a finite set of examples

Elements

1. a set of labeled examples :

$$\left(\mathbf{x}^{(i)}, y^{(i)}\right)_{i=1}^{N}$$

2. an evaluation measure $\Rightarrow$ to evaluate « progresses »
3. a class of hypotheses
    - « structure » of the program we want to learn
    - cf. inductive bias

Evaluation

Context

- goal : which of two classifiers is the best ? (or the same classifier during training)
- need an automatic, quantitative metric
- main idea : compare the output of the program (i.e. predicted value) to the expected output

same as unit test

Loss function (1)

- estimate the cost of a bad prediction
- formally : $\ell(\hat{y}, y^*) : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ where :
  - $\hat{y}$ gold label (given by an oracle)
  - y^* predicted label
- by definition : $\ell(y, y) = 0$, the larger ℓ the worse the decision ;
- regression : $\ell_{\text{RSE}} = (y^* - \hat{y})^2$
- binary classification :

$$\begin{aligned} \ell_{0/1} &= \mathbb{1}_{\hat{y}=y^*} \\ &= \begin{cases} 1 & \text{if } \hat{y} \neq y^* \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Loss function (2)

- new task : detect if somebody is sick (binary classification)
- two kinds of error :
  1. false negative : ‘miss’ that the patient is sick
  2. false positive : predict somebody is sick when she is not
- case №1. is ‘worst’ than case №2... but $\ell^{0/1}$ is symmetric
- weighted loss :

predicted	gold	loss
positive	positive	0
	negative	γ_1
negative	positive	γ_2
	negative	0

γ_1 and γ_2 are problem-dependent

Evaluation Principle

A classifier is evaluated by the mean of the loss function over a dataset :

$$\begin{aligned} \mathcal{E} &= \frac{1}{n} \sum_i \ell\left(\hat{y}^{(i)}, y^{(i)*}\right) \\ &= \frac{1}{n} \sum_i \ell\left(f(\mathbf{x}^{(i)}), y^{(i)*}\right) \end{aligned}$$

- the smaller the *(error rate)*, the better the classifier

Different kind of errors

- training error :
  - estimated on the examples used for training
  - evaluate how well the classifier ‘fit’ the data
- test error :
  - estimated on an independent dataset
  - evaluate how well a the classifier will ‘generalize’
  - “close” to the error we will achieve on any other dataset (cf. section on theoretical guarantees)

Linear classifiers

Warning

For the moment : only binary classification : $y = \pm 1$

Supervised classification framework

Building blocks of supervised classification

1. labeled examples ;
2. loss function ;
3. hypotheses class.

Hypotheses class

- decision rule : how to choose the label associated to an observation
- learning algorithm : choose one element in this class

Recall : decision tree & k-nn

Decision Tree

- make decision by considering a small number of features
- conjunction of features

k-nn

- many features but they are all considered equal
- curse of dimensionality

Neither of these two extremes is desirable

The class of linear function

Scoring function

$$F(\mathbf{x}; \mathbf{w}) = \mathbf{w} \cdot \mathbf{x}$$

- feature vector usually have a constant feature (bias)
- function parametrized by a parameter vector $\mathbf{w}$
- linear combination of features : each features has a weight describing its importance

Decision function

$$y^* = \text{sign}\, F(\mathbf{x}; \mathbf{w}) = \begin{cases} 1 & \text{if } F(\mathbf{x}; \mathbf{w}) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Recall : dot product

- the dot product of $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^n$ is a real :

$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i \times y_i \tag{1}$$

- in python :

```
np.dot(x, y) # x and y and numpy arrays
sum(xx * yy for xx, yy in zip(x, y)) # x and y are lists (same siz
```

- interpretation : weighted sum
  - assume $\mathbf{x}$ is a binary vector (e.g. the i-th word is present or not)
  - y_i quantifies the importance of the i-th word in the total score
    - y_i 'small' $\Rightarrow$ word has no impact on overall score
    - y_i 'large' $\Rightarrow$ i-th word may change the sign !

Decision frontier

Definition

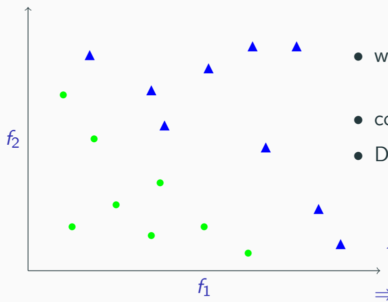
- characterize the class of functions
- = when the decision of the decision function changes

In our case...

$$F(\mathbf{x}; \mathbf{w}) = 0$$

$\Rightarrow$ equation of an hyperplane = separating hyperplane

Graphically

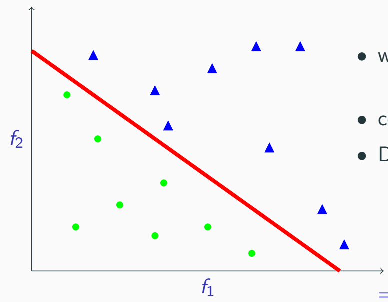


- with 3 features : $\mathbf{x} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$
- constant feature = bias
- Decision frontier :

$$\mathbf{x} \cdot \mathbf{w} = 0$$
$$x_1 \cdot w_1 + x_2 \cdot w_2 + w_0 = 0$$

$\Rightarrow$ equation of line

Graphically

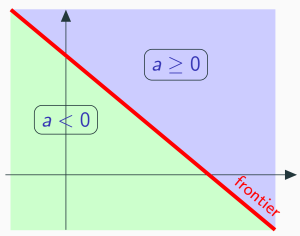


- with 3 features : $\mathbf{x} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$
- constant feature = bias
- Decision frontier :

$$\mathbf{x} \cdot \mathbf{w} = 0$$
$$x_1 \cdot w_1 + x_2 \cdot w_2 + w_0 = 0$$

$\Rightarrow$ equation of line

Graphically



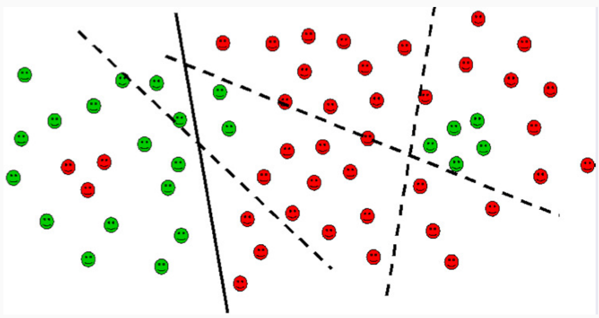
- with 3 features : $\mathbf{x} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$
- constant feature = bias
- Decision frontier :

$$\mathbf{x} \cdot \mathbf{w} = 0$$
$$x_1 \cdot w_1 + x_2 \cdot w_2 + w_0 = 0$$

$\Rightarrow$ equation of line

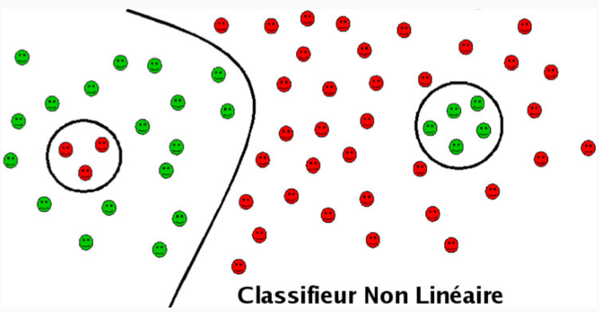
- decision function

Expressivity



Datasets are not all linearly separable

Non-linear classification



either Kernel of Multi-Layer Neural Networks

Learning

How to choose $\mathbf{w}$?

Goal

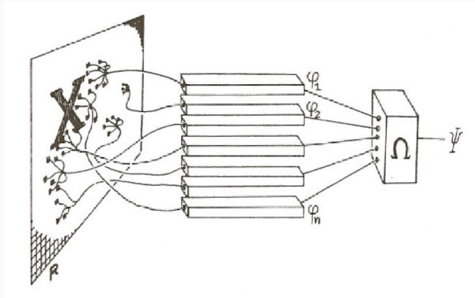
- make as few errors as possible on new examples
- must be chosen from a finite dataset (training set)

Methods

- perceptron
- maxium entropy classifier (Maxent)
- support vector machines (SVM)
- Winnow
- ...

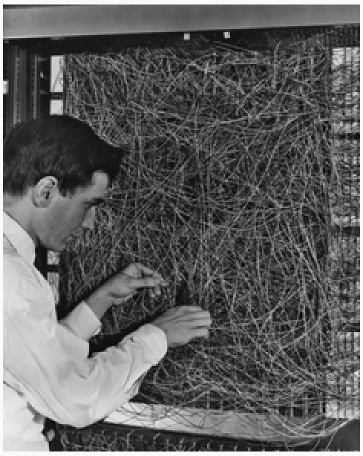
Perceptron

History (1)



- Rosenblatt, 1957
- goal = simulate visual capacity of human brain
- has motivated both learning algorithm & parametrization... not useful anymore

The perceptron is a machine !



Learning

Learning through error correcting (Rosenblatt, 57)

- for each example of the training set
- predict its label
- if the label is correct : do nothing
- if the label is not correct : correct parameters to predict the correct label

More formally

Require: a training set $\left(\mathbf{x}^{(i)}, y^{(i)}\right)_{i=1}^{n}$

```
w ← 0
while there are classification errors do
    for i = 1 to n do
        y = sign(w · x^(i))
        if y ≠ y^(i) then
            w ← w + ·y^(i) · x^(i)
        end if
    end for
end while
```


Learning through error correcting

Update rule :

$$\mathbf{w_{t+1}} \leftarrow \mathbf{w_t} + \cdot y^{(i)} \cdot \mathbf{x}^{(i)}$$

- only when an error occurs
- let us assume that $y^{(i)} = 1$ (same reasoning with $y^{(i)} = -1$)
  - the scoring function is negative, even though it should have been positive
  - should increase its « value » for the example at stake
- in practice :

$$\begin{aligned} F(\mathbf{x}; \mathbf{w}_{t+1}) - F(\mathbf{x}; \mathbf{w}_t) &= \mathbf{x} \cdot \mathbf{w}_{t+1} - \mathbf{w}_t \\ &= y^{(i)} \times \mathbf{x} \cdot \mathbf{x} \\ &= y^{(i)} \times \underbrace{||x||_2^2}_{\geq 0} \end{aligned}$$

Theoretical results

- if the training set is linearly separable :
  - converges in a finite number of steps
  - no guarantee about the generalization error
- otherwise :
  - infinite loop
  - several tricks, no proof
- how to know if a training set is linearly separable ? $\Rightarrow$ you have to try !

Why does it work ?

What we did...

- choose $\mathbf{w}$ so that there is no error on the training set

What we are interested in...

- prediction quality on unknown examples ?

Link ?

Improving the perceptron

Menu

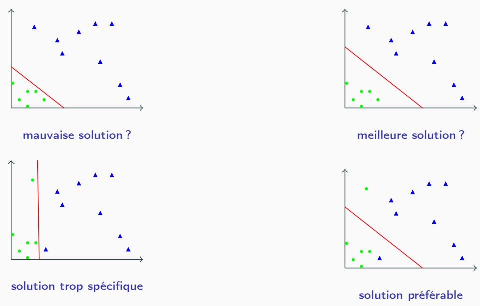
- what to do with non-linearly separable datasets ?
- stability issues

What to do with non-linearly separable data ?

- Heuristic to optimize $\ell^{0/1}$ on the train set
- no theoretical guarantee on the convergence (only for linearly separable dataset
- no guarantee on the generalization performance

Limits of the perceptron

Do we known anything about the generalization error ?



Thanks to Mathematics...

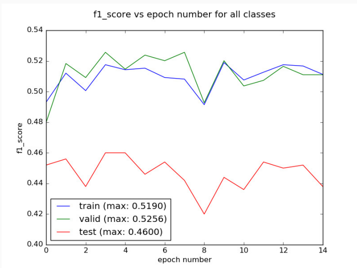
- no connection between learning error and generalization error (for the perceptron)
- but, in practice : experimental estimation of the generalization error
- use 3 independent datasets :
  1. training set : to estimate the parameters of the perceptron
  2. validation set : to monitor the generalization error
     - regularly, compute the error on the validation set
     - keep the value of the parameters with the lowest error
  3. test set : to estimate the generalization error (can only be used once)

More formally

Require: a training set $\left(\mathbf{x}^{(i)}, y^{(i)}\right)_{i=1}^{n}$, hyper-parameter MAX_EPOCH

$\mathbf{w} \leftarrow 0$
for $epoch = 1$ to MAX_EPOCH **do**
  shuffle training set
  for $i = 1$ to n **do**
    $y = \text{sign}\left(\mathbf{w} \cdot \mathbf{x}^{(i)}\right)$
    if $y \neq y^{(i)}$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + \cdot y^{(i)} \cdot \mathbf{x}^{(i)}$
    end if
  end for
  compute error on validation set
end for

With a picture...



- epoch = iteration on the whole training set
- learning curve
- You should NEVER draw this curve : use the test set only ONCE (here : pedagogical purpose)

Why does it work ?

- unbiased estimation of the generalization error (see central limit theorem)
- error rate is a mean $\Rightarrow$ can be estimated from a sample if this sample is large enough
- but : we need more labeled data + computational cost of evaluating the error

Stability issues

The problem

- 10,000 examples, after the 100-th we found ‘good’ parameters $\rightarrow$ no errors on the following 9 899-th examples...
- error when predicting the label of the 10,000-th example $\Rightarrow$ the update will ‘destroy’ parameters that we very good for 99.99% of the examples

Solution

- `random.shuffle`
- averaged perceptron

Voted Perceptron

Principle

- keep track of $\mathbf{w}$ at each step
- consider one classifier for each value of $\mathbf{w}$
- majority vote of all classifiers

Pros/cons

- theoretical results : generalization error of the voted perceptron is always lower than the one of the ‘vanilla’ perceptron
- cons : computational cost

Averaged Perceptron

- approximation of the *voted perceptron*
- instead of considering the last value of the parameter vector, we consider the mean of all the values it has taken during learning
- no more theoretical results, but works experimentally

Algorithm

Require: a training set $\left(\mathbf{x}^{(i)}, y^{(i)}\right)_{i=1}^{n}$

$\mathbf{w} \leftarrow 0$
$\mathbf{a} \leftarrow 0$
while there are classification errors **do**
  for $i = 1$ to n **do**
    $y = \text{sign}\left(\mathbf{w} \cdot \mathbf{x}^{(i)}\right)$
    if $y \neq y^{(i)}$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + \cdot y^{(i)} \cdot \mathbf{x}^{(i)}$
    end if
    $\mathbf{a} \leftarrow \mathbf{a} + \mathbf{w}$
  end for
end while

Why does it work ?

- 3 iterations of the training algorithm $\Rightarrow$ 3 weights :

$$\begin{array}{c|c} \mathbf{w}^1 & \mathbf{x} \cdot \mathbf{w}^1 > 0 \\ \mathbf{w}^2 & \mathbf{x} \cdot \mathbf{w}^2 < 0 \\ \mathbf{w}^3 & \mathbf{x} \cdot \mathbf{w}^3 > 0 \end{array}$$

- voted perceptron : $\oplus$ (two votes versus one)
- average perceptron :

$$\mathbf{x} \cdot \mathbf{w}^a = \frac{1}{3} \times \left(\underbrace{\mathbf{x} \cdot \mathbf{w}^1}_{>0} + \underbrace{\mathbf{x} \cdot \mathbf{w}^2}_{<0} + \underbrace{\mathbf{x} \cdot \mathbf{w}^3}_{>0} \right) \tag{2}$$

if $\mathbf{x} \cdot \mathbf{w}^2$ is not too 'large', the average score will be positive

Multi-class classification

The setting

- each observation $\mathbf{x}$ is associated to a label $y \in \mathcal{Y}$
- $\mathcal{Y}$ = finite set of possible labels (in practice $[\![1, K]\!]$) ($\mathcal{Y}$ can contain up to several thousands labels)
- multi-class loss :

$$\ell^{\text{multi}}(y_1, y_2) = \begin{cases} 0 & \text{if } y_1 = y_2 \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

Decision function

1st formulation

- one weight vector for each class $\mathbf{w}_y$
- decision :

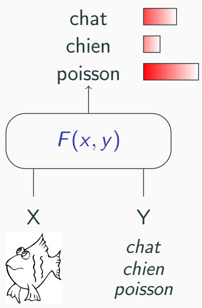
$$y^* = \arg\max_{y \in \mathcal{Y}} \mathbf{x} \cdot \mathbf{w}_y \tag{4}$$

2nd formulation

- single weight vector $\mathbf{w}$
- a feature f is expanded into K features : $f \text{ AND } y$

$$y^* = \arg\max_{y \in \mathcal{Y}} \underbrace{\phi(\mathbf{x}, y)}_{\text{joint representation of } \mathbf{x} \text{and } y} \cdot \mathbf{w} \tag{5}$$

Interpretation



Training algorithm

- as before
- new update rule (if $y^* \neq y^{(i)}$) :

$$\mathbf{w} \leftarrow \mathbf{w} + \phi\left(\mathbf{x}, y^{(i)}\right) - \phi\left(\mathbf{x}, y^*\right) \tag{6}$$

- 'penalize' the score of the label that has been incorrectly predicted, 'increase' the score of the correct label