Supervised methods

Naïve Bayes and Perceptron

Dr. Jakub Kuzilek

Introduction

Introduction

Today you will learn:

- Naïve Bayes classifier
- Perceptron

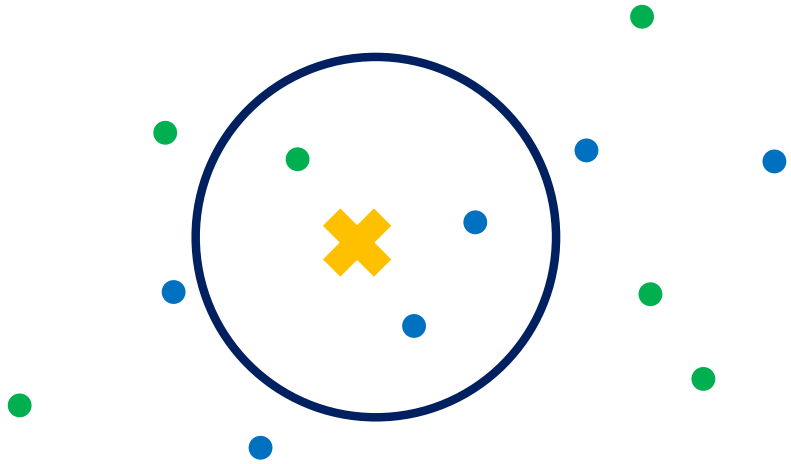
Introduction

Small recap: Can you explain k-Nearest Neighbours algorithm?

Introduction

Small recap: Can you explain k-Nearest Neighbours algorithm?

3-NN



Naïve Bayes classifier

Bayes' rule

$$p(A, B)$$

Bayes' rule

$$p(A, B)$$

… **joint probability** of observing events A and B happening together

Bayes' rule

$$p(A,B) = p(A|B)p(B) = p(B|A)p(A)$$

$p(A|B)$ and $p(B|A)$ are **conditional probabilities** of observing event A (B respectively) when event B (A respectively) happens. $p(A)$ and $p(B)$ are probabilities of events A and B happening without regard to each other

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

Bayesian decision problem

- Given:
  - $X, Y, \hat{Y}$
  - $p_{XY}: X \times Y \rightarrow \mathbb{R}$ the joint probability that the sample is in state y and the measurement x is made
  - $W: Y \times \hat{Y} \rightarrow \mathbb{R}$ is loss function that $W(y, \hat{y}), y \in Y, \hat{y} \in \hat{Y}$ is penalty paid for the object in state y and the decision $\hat{y}$ is made
- Then for strategy $f: X \rightarrow \hat{Y}$ we can compute the expectation of $W(y, f(x))$ as:

$$R(f) = \sum_{x \in X} \sum_{y \in Y} p_{XY}(x, y) W\big(y, f(x)\big),$$

where $R(f)$ is called Bayesian risk

Bayesian decision problem

- We aim at finding optimal strategy f^*, which minimizes the Bayesian risk:

$$f^* = \underset{f \in X \to \hat{Y}}{\operatorname{argmin}} R(f)$$

- Then $R(f^*)$ is:

$$\begin{aligned} R(f^*) &= \min_{f \in X \to \hat{Y}} \sum_{x \in X} \sum_{y \in Y} p_{XY}(x,y) W\big(y, f(x)\big) \\ &= \sum_{x \in X} \min_{f(x) \in \hat{Y}} \sum_{y \in Y} p_{XY}(x,y) W\big(y, f(x)\big) \\ &= \sum_{x \in X} p(x) \min_{f(x) \in \hat{Y}} \sum_{y \in Y} p_{Yx}(y|x) W\big(y, f(x)\big) \\ &= \sum_{x \in X} p(x) \min_{f(x) \in \hat{Y}} R(x, \hat{y}) \end{aligned}$$

- The expectation of loss conditioned by x is called partial risk:

$$R(x, \hat{y}) = \sum_{y \in Y} p_{Yx}(y|x) W(y, \hat{y}\,)$$

Bayesian decision problem

- From $R(f^*) = \sum_{x \in X} p(x) \min_{f(x) \in \hat{Y}} R(x, \hat{y})$ follows that the minimization of Bayesian risk can be achieved by minimization of partial risk for each x.

- Optimal strategy f^*:

$$f^* = \operatorname*{argmin}_{\hat{y} \in \hat{Y}} \sum_{y \in Y} p_{Yx}(y|x) W(y, \hat{y})$$

Naïve Bayes classifier

- Classification with 0-1 loss function:

$$W\big(y, f(x)\big) = \begin{cases} 0, & if\ f(x) = y \\ 1, & if\ f(x) \neq y \end{cases}$$

- Partial risk for x:

$$R\big(x, f(x)\big) = \sum_{y \in Y} p_{Yx}(y|x) W(y, f(x)\ )$$
$$= \sum_{y \neq f(x)} p_{Yx}(y|x)$$
$$= 1 - p_{Yx}(f(x)|x)$$

- Optimal strategy f^*:

$$f^* = \underset{f(x) \in \hat{Y}}{\operatorname{argmin}}\, R(x, y) = \underset{f(x) \in \hat{Y}}{\operatorname{argmax}}\, p_{Yx}(f(x)|x) = \underset{f(x) \in \hat{Y}}{\operatorname{argmax}}\, p_{Yx}(\hat{y}|x)$$

posterior probability

Naïve Bayes classifier

- Please note that x is a vector of features $x = (x_1, x_2, \dots, x_M)$, where M is number of features
- Thus:

$$p_{Yx}(y|x) = p_{Yx}(y|x_1, x_2, \dots, x_M)$$

- When M is larger or x_i have large number of values the model estimation is infeasible
- We can rewrite:

prior probability of y

likelihood of x being in y

$$p_{Yx}(y|x) = \frac{p_Y(y)p_{Yx}(x|y)}{p_X(x)} = \frac{p_{xY}(y,x)}{p_X(x)}$$

evidence of x

Naïve Bayes classifier

- In practice only nominator is interesting
- Denominator is effectively constant (features are given)
- Using chain rule:

$$
\begin{aligned}
p_{xY}(y, x) &= p_{xY}(y, x_1, x_2, \dots, x_M) \\
&= p_{xY}(x_1, x_2, \dots, x_M, y) \\
&= p_{xY}(x_1 | x_2, \dots, x_M, y) p_{xY}(x_2, \dots, x_M, y) \\
&= \cdots \\
&= p_{xY}(x_1 | x_2, \dots, x_M, y) p_{xY}(x_2 | x_3, \dots, x_M, y) \\
&\dots p_{xY}(x_{M-1} | x_M, y) p_{xY}(x_M, y) p_Y(y)
\end{aligned}
$$

Naïve Bayes classifier

- And here comes the naïve part: **All features in vector x are conditionally independent.**
- Thus:

$$p_{xY}(x_i|x_{i+1}, \dots, x_M, y) = p_{xY}(x_i|y)$$

- And the joint model can be expressed as:

$$p_{xY}(\widehat{y}|x) \propto p_{xY}(y, x)$$
$$\propto p_Y(y)p_{xY}(x_1|y)p_{xY}(x_2|y) \dots p_{xY}(x_M|y)$$
$$\propto p_Y(y) \prod_{i=1}^{M} p_{xY}(x_i|y)$$

Example

- Students are taking test and can have three possible outcomes: $X = \{0, 0.5, 1\}$. Based on the test result predict if the student will $Y = \hat{Y} = \{pass, fail\}$. $N = 100$

Example

	1	0.5	0	Total
passed	60	30	10	100
failed	0	30	70	100
Total	60	60	80	200

- Priors:

$$p(passed) = \frac{100}{200} = 0.5$$

$$p(failed) = \frac{100}{200} = 0.5$$

Example

	1	0.5	0	Total
passed	60	30	10	100
failed	0	30	70	100
Total	60	60	80	200

- Evidence:

$$p(1) = \frac{60}{200} = 0.3$$

$$p(0.5) = \frac{60}{200} = 0.3$$

$$p(0) = \frac{80}{200} = 0.4$$

Example

	1	0.5	0	Total
passed	60	30	10	100
failed	0	30	70	100
Total	60	60	80	200

- Likelihood:

$$p(1|passed) = \frac{60}{100} = 0.6$$

$$p(1|failed) = \frac{0}{100} = 0$$

Example

	1	0.5	0	Total
passed	60	30	10	100
failed	0	30	70	100
Total	60	60	80	200

- Likelihood:

$$p(0.5|passed) = \frac{30}{100} = 0.3$$

$$p(0.5|failed) = \frac{30}{100} = 0.3$$

Example

	1	0.5	0	Total
passed	60	30	10	100
failed	0	30	70	100
Total	60	60	80	200

- Likelihood:

$$p(0|passed) = \frac{10}{100} = 0.1$$

$$p(0|failed) = \frac{70}{100} = 0.7$$

Example

	1	0.5	0	Total
passed	60	30	10	100
failed	0	30	70	100
Total	60	60	80	200

- Posterior probability:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

$p(y\|x)$	1	0.5	0
passed	**1** ($\frac{0.6*0.5}{0.3}$)	0.5	0.125
failed	0	**0.5**	**0.875**

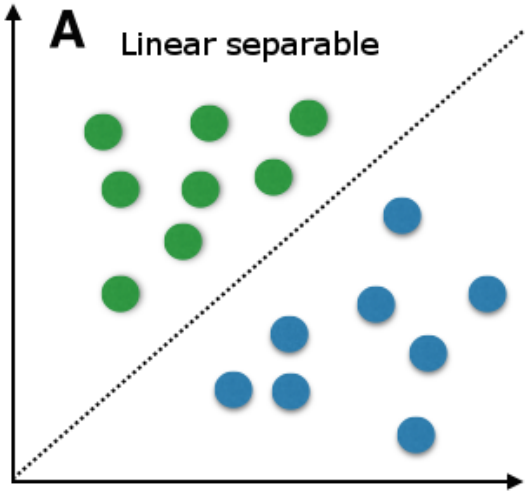
Perceptron

Linear Classifier & Linearly separable data

- Let $x = (x_1, x_2, \ldots, x_M)$ is the set of features
- Then the linear classifier is the decision function f in form of:

$$f(x) = w_1 x_1 + w_2 x_2 + \cdots + w_M x_M + w_0 = \langle w, x \rangle + w_0$$

- The linear classifier represents the linear decision boundary (separating hyperplane)



https://leonardoaraujosantos.gitbook.io/artificial-inteligence/machine_learning/supervised_learning/linear_classification

Algorithm

- Training set:

$$T = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_N, y_N)\},$$

where $x_j = (x_1, x_2, \dots, x_M, 1)$ and $y_j \in \{-1, 1\}$

- We are searching for set of weights $w = (w_1, w_2, \dots, w_M, w_0)$, which satisfies:

$$\langle w, x_j \rangle \geq 0, \qquad for\ y_j = \ \ 1$$
$$\langle w, x_j \rangle < 0, \qquad for\ y_j = -1$$

- We can rewrite both conditions to one:

$$\langle w, y_j x_j \rangle \geq 0$$

Algorithm

- Algorithm:
  1. Set all weights to 0: $w_0 = (w_1, w_2, \dots, w_M, w_0) = (0, 0, \dots, 0)$
  2. For each sample x_j do:
     - Determine if the sample is classified correctly $\langle w, y_j x_j \rangle \geq 0$
     - If yes go to the next sample
     - If not update weights

$$w_{i+1} = w_i + x_j$$

  3. Repeat 2. until the error $\frac{1}{N} \sum_{j=1}^{N} |y_j - sign(\langle w, x_j \rangle)|$ is bellow threshold γ or number of iterations i is reached

Properties & Extensions

- If the data is **linearly separable** then the algorithm will find correct classification and ends in non-infinite number of steps (Novikoff's theorem)

- It does not converge on non-linearly separable data
- In order to work on the non-linearly separable data we can transform data to higher dimension

- Example – polynomial transformation:

$$x = (x_1, x_2)$$

$$\hat{x} = (x_1, x_2, x_1^2, x_2^2, x_1 x_2)$$

Questions?