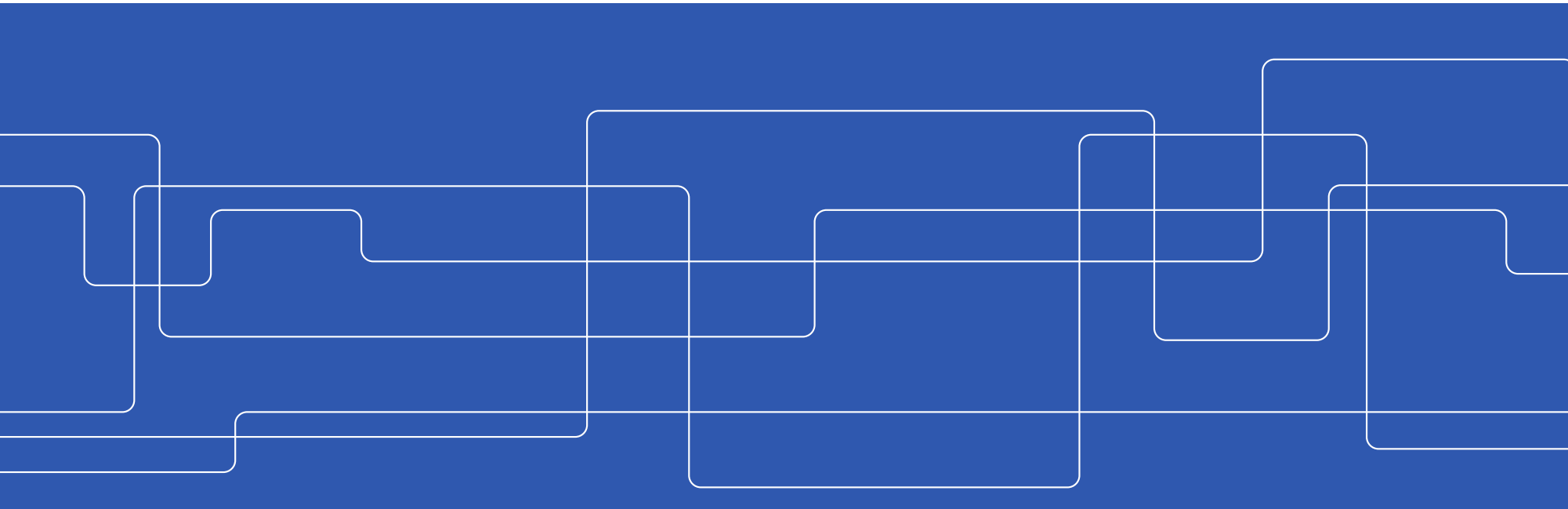
Programming for Data Science – Naïve Bayes and k-nearest neighbors

Henrik Boström

Prof. of Computer Science - Data Science Systems
Division of Software and Computer Systems
Department of Computer Science
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology
bostromh@kth.se



Outline

The restaurant example

Ex.	Other	Bar	Fri/Sat	Hungry	Guests	Wait
e1	yes	no	no	yes	some	yes
e2	yes	no	no	yes	full	no
e3	no	yes	no	no	some	yes
e4	yes	no	yes	yes	full	yes
e5	yes	no	yes	no	none	no
e6	no	yes	no	yes	some	yes

$$P(Wait_{yes} | Hungry_{yes} \& Guests_{full} \& Bar_{no} \& \ldots)$$
$$P(Wait_{no} | Hungry_{yes} \& Guests_{full} \& Bar_{no} \& \ldots)$$

Bayes' theorem

$$P(H|E) = \frac{P(H\&E)}{P(E)} = \frac{P(H)P(H\&E)}{P(E)P(H)} = \frac{P(H)P(E|H)}{P(E)}$$

$$P(H|E) = \frac{P(H)P(E|H)}{P(E)}$$

Bayes' theorem (example)

$$P(c|x_1 \& \ldots \& x_m) = \frac{P(c)P(x_1 \& \ldots \& x_m|c)}{P(x_1 \& \ldots \& x_m)}$$

$$P(Wait_{yes}|Hungry_{yes} \& Guests_{full} \& Bar_{no} \& \ldots) =$$

$$\frac{P(Wait_{yes})P(Hungry_{yes} \& Guests_{full} \& Bar_{no} \& \ldots|Wait_{yes})}{P(Hungry_{yes} \& Guests_{full} \& Bar_{no} \& \ldots)}$$

Naïve Bayes

The "naïve" assumption of conditional independence:

$$P(x_1 \& \ldots \& x_m | c) = P(x_1|c) \cdots P(x_m|c)$$

$$P(Wait_{yes} | Hungry_{yes} \& Guests_{full} \& Bar_{no} \& \ldots) =$$

$$\frac{P(Wait_{yes}) P(Hungry_{yes} | Wait_{yes}) P(Guests_{full} | Wait_{yes}) P(Bar_{no} | Wait_{yes}) \ldots}{P(Hungry_{yes} \& Guests_{full} \& Bar_{no} \& \ldots)}$$

The restaurant example

Ex.	Other	Bar	Fri/Sat	Hungry	Guests	Wait
e1	yes	no	no	yes	some	yes
e2	yes	no	no	yes	full	no
e3	no	yes	no	no	some	yes
e4	yes	no	yes	yes	full	yes
e5	yes	no	yes	no	none	no
e6	no	yes	no	yes	some	yes

$P(Wait_{yes})$ $P(Hungry_{yes}|Wait_{yes})$ $P(Guests_{full}|Wait_{yes})$ $P(Bar_{no}|Wait_{yes})$

$4/6$ $3/4$ $1/4$ $2/4$

$P(Wait_{no})$ $P(Hungry_{yes}|Wait_{no})$ $P(Guests_{full}|Wait_{no})$ $P(Bar_{no}|Wait_{no})$

$2/6$ $1/2$ $1/2$ $2/2$

The restaurant example (cont.)

$P(Wait_{yes})$ $P(Hungry_{yes}|Wait_{yes})$ $P(Guests_{full}|Wait_{yes})$ $P(Bar_{no}|Wait_{yes})$

$$4/6 \times \quad 3/4 \times \quad 1/4 \times \quad 2/4 \quad = 1/16$$

$P(Wait_{no})$ $P(Hungry_{yes}|Wait_{no})$ $P(Guests_{full}|Wait_{no})$ $P(Bar_{no}|Wait_{no})$

$$2/6 \times \quad 1/2 \times \quad 1/2 \times \quad 2/2 \quad = 1/12$$

$$P(Wait_{yes}|Hungry_{yes} \& Guests_{full} \& Bar_{no}) = 3/7$$
$$P(Wait_{no}|Hungry_{yes} \& Guests_{full} \& Bar_{no}) = 4/7$$

The tic-tac-toe example

S1 **X**	S2 **X**	S3 **X**
S4 **B**	S5 **O**	S6 **B**
S7 **B**	S8 **B**	S9 **O**

$$P(-|S1_X \& S2_X \& S3_X) = P(-)\frac{P(S1_X \& S2_X \& S3_X|-)}{p(S1_X \& S2_X \& S3_X)}$$

$$P(S1_X \& S2_X \& S3_X|-) = P(S1_X|-)P(S2_X|-)P(S3_X|-)$$

The tic-tac-toe example (cont.)

+	S_1	S_2
X	25	20
O	18	20
B	0	3

-	S_1	S_2
X	15	25
O	20	20
B	10	0

$$P(+|S1_B \& S2_B \& \ldots) = P(+)P(S1_B|+)\ldots = 43/88 \times 0/43\ldots$$
$$P(-|S1_B \& S2_B \& \ldots) = P(-)P(S1_B|-)P(S2_B|-)\ldots = 45/88 \times 10/45 \times 0/45\ldots$$

Naïve Bayes in practice

- What if $P(x_v|c) = 0$?
  - Laplace correction may be employed, i.e., $P(x_v|c) = \frac{n+1}{m+k}$, where n is the number of observations of x_v when c is present, m is the number of observations of x_w for any value w when c is present and k is the number of possible values for x.

The tic-tac-toe example with Laplace correction

+	S_1	S_2
X	25+1	20+1
O	18+1	20+1
B	0+1	3+1

-	S_1	S_2
X	15+1	25+1
O	20+1	20+1
B	10+1	0+1

$$P(+|S1_B \& S2_B \& \ldots) = P(+)P(S1_B|+)\ldots = 43/88 \times 1/46\ldots$$
$$P(-|S1_B \& S2_B \& \ldots) = P(-)P(S1_B|-)P(S2_B|-)\ldots = 45/88 \times 11/48 \times 1/48\ldots$$

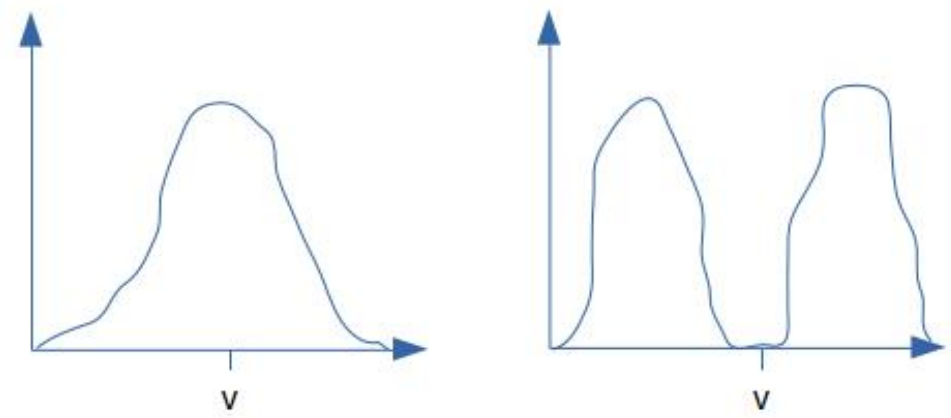
Naïve Bayes in practice (cont.)

- What if some feature value is missing for an instance?
  - For a test instance: ignore the feature when calculating class probabilities
  - For a training instance: ignore the feature when updating counts

Naïve Bayes in practice (cont.)

- What if some feature is numerical?
  - Employ discretization (binning), or
  - Use a probability density function, e.g.,
    $P(v - \epsilon/2 \leq x \leq v + \epsilon/2|c) \approx \epsilon f(v, \mu_{x,c}, \sigma_{x,c})$
    Note that for Naïve Bayes, ϵ is cancelled out.

Handling numerical features with Naïve Bayes



$$f(v, \mu_{x,c}, \sigma_{x,c})$$

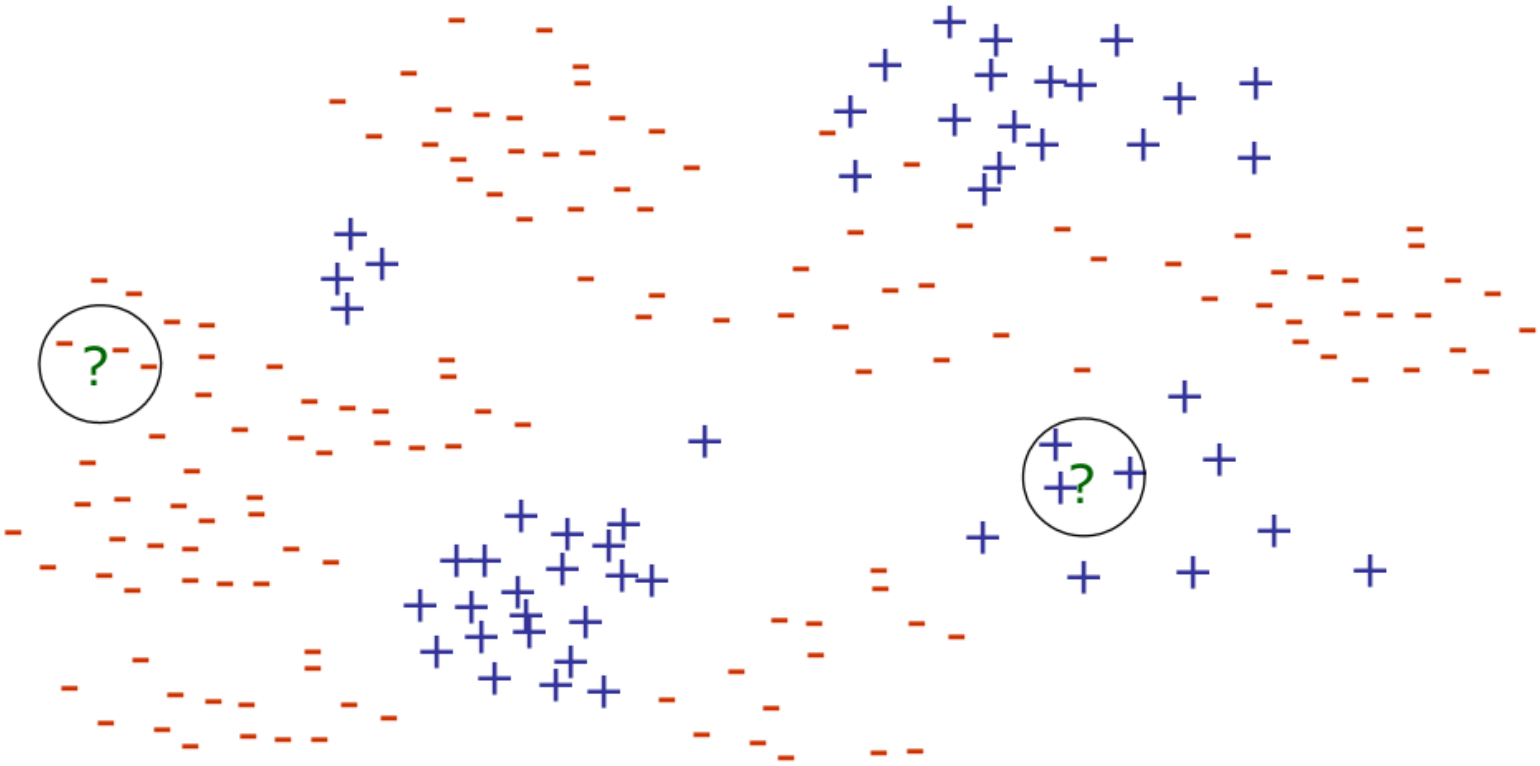
Naïve Bayes in practice (cont.)

- Can we interpret the model/understand the predictions?
  - To be discussed ...

Implementing Naïve Bayes

- The learning algorithm is very simple; we mainly have to keep counts for each class label and possible feature value.
- Note that we need to record all transformations on training data, e.g., bins, so that these can be employed also on test data, before making predictions.
- During prediction, calculating the *log probability* is often recommended for numerical stability, i.e., performing addition instead of multiplication, in particular for large numbers of features
- Implementing both *batch learning*, i.e., assuming that we have all training data from the start, and *incremental learning*, i.e., updating the model after each new incoming instance, are quite straightforward (unless transformations are needed).

k-nearest neighbors (lazy learning)



The k-nearest neighbors algorithm

```
Input: test instance e, training examples E, constant k
Output: class label c

Let N be the k closest instances to e in E

Let c be the majority class of N
```

k-nearest neighbors in practice

- A suitable distance metric has to be chosen; a common choice being the Euclidean distance

$$d(X_1, X_2) = \sqrt{(X_1 - X_2)^2}$$

- The Euclidean distance metric requires that
  - categorical features are converted to numerical
  - missing values are imputed
  - numerical features are normalized

k-nearest neighbors in practice (cont.)

- Can we interpret the model/understand the predictions?
  - To be discussed ...

k-nearest neighbors in practice (cont.)

- The size of the model grows with the number of training instances, which may prevent the algorithm from being used in resource-constrained environments
- The computational bottleneck is during prediction (inference), as each test instance requires distance calculations for all training instances
- Approaches to speeding up the algorithm include
  - reducing dimensionality
  - sampling training data or prototype selection
  - partitioning the feature space, e.g., by k-d trees

k-nearest neighbors extensions

- The algorithm can be easily adapted to regression tasks (numerical prediction), e.g., by averaging the predictions of the nearest neighbors.
- The distance metric may take feature weights into account, e.g.,

$$d(X_1, X_2) = \sqrt{(w(X_1 - X_2))^2}$$

- The voting procedure may take distances into account, e.g.,

$$w_i = \begin{cases} \frac{d_k - d_i}{d_k - d_1}, & \text{if } d_1 \neq d_k. \\ 1, & \text{otherwise.} \end{cases}$$

Implementing k-nearest neighbors

- The learning algorithm is the simplest possible; we just have to remember all observations
- Again, we need to record all data transformation steps, e.g., one-hot encodings, normalizations, etc., so that these can be employed also on test data, before making predictions.
- Implementing batch learning is straightforward, while an incremental implementation would require that the data transformation procedure is continuously updated.

Summary

- Two basic learning algorithms have been considered; naïve Bayes and the k-nearest Neighbor algorithm.
- They are among the fastest algorithms during training. However, there is a substantial computational cost associated with making predictions using kNN.
- We have seen what requirements the algorithms have on data transformations as well as various ways of extending the algorithms.