

Machine Intelligence

9. Supervised Learning Part III: Other Algorithms and Evaluation

Álvaro Torralba



AALBORG UNIVERSITET

Fall 2022

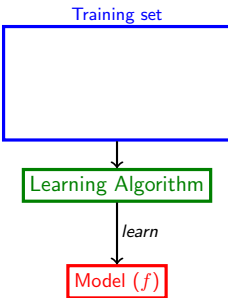
Thanks to Thomas D. Nielsen and Jörg Hoffmann for slide sources

Agenda

- 1 Introduction
- 2 Naive Bayes: Using Probabilistic Models
- 3 Case-based Reasoning
- 4 Generalization
- 5 Overfitting
- 6 Training ML Models To Avoid Overfitting
- 7 Evaluating the Quality of Classification Models
- 8 Conclusion

In previous chapters of the MI Lecture... Supervised Learning

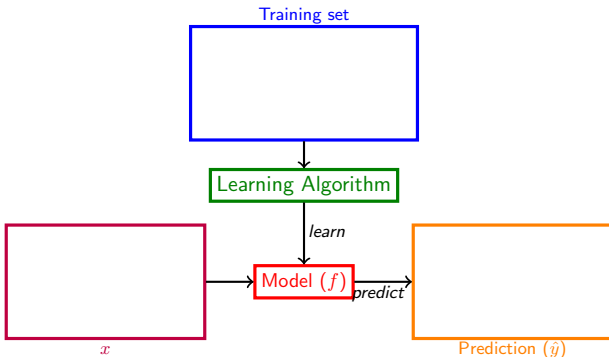
- First step: Learning from training data
 - Input: **Training set**
 - Output: **Model (f)**



- Each example is a pair $(x, f(x))$ for some unknown function
- The learning algorithm returns a function that approximates f for a collection of examples

In previous chapters of the MI Lecture... Supervised Learning

- First step: Learning from training data
 - Input: **Training set**
 - Output: **Model (f)**
- Second Step: Predicting the target feature using the learned model (f)
 - Input: **New data (x)**
 - Output: **Prediction feature (y)**



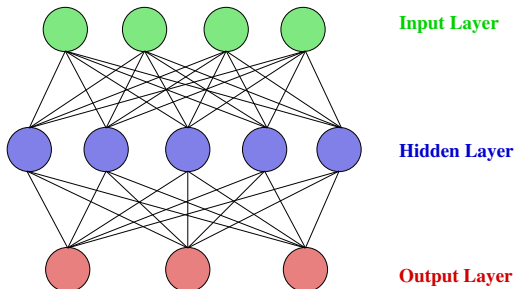
- Each example is a pair $(x, f(x))$ for some unknown function
- The learning algorithm returns a function that approximates f for a collection of examples

In previous chapters of the MI Lecture...

Decision Trees



Neural Networks



Our Agenda for this Chapter

- **Supervised Learning using probabilistic models.**
 - **How to use Bayesian Networks for learning.**

Our Agenda for this Chapter

- **Supervised Learning using probabilistic models.**
 - **How to use Bayesian Networks for learning.**
- **Case-based Reasoning:** is there any alternative to Neural Networks?
 - **Yet another alternative for supervised learning**

Our Agenda for this Chapter

- **Supervised Learning using probabilistic models.**
 - **How to use Bayesian Networks for learning.**
- **Case-based Reasoning:** is there any alternative to Neural Networks?
 - **Yet another alternative for supervised learning**
- **Overfitting:** What happens when learning too much?
 - **A common problem in supervised learning.**

Our Agenda for this Chapter

- **Supervised Learning using probabilistic models.**
 - **How to use Bayesian Networks for learning.**
- **Case-based Reasoning:** is there any alternative to Neural Networks?
 - **Yet another alternative for supervised learning**
- **Overfitting:** What happens when learning too much?
 - **A common problem in supervised learning.**
- **Training ML Models:** How to avoid overfitting?
 - **How to split data into train and test set.**

Our Agenda for this Chapter

- **Supervised Learning using probabilistic models.**
 - **How to use Bayesian Networks for learning.**
- **Case-based Reasoning:** is there any alternative to Neural Networks?
 - **Yet another alternative for supervised learning**
- **Overfitting:** What happens when learning too much?
 - **A common problem in supervised learning.**
- **Training ML Models:** How to avoid overfitting?
 - **How to split data into train and test set.**
- **Evaluating the Quality of Classification Models:**
 - **Can we trust the learned models?**

Agenda

- 1 Introduction
- 2 Naive Bayes: Using Probabilistic Models
- 3 Case-based Reasoning
- 4 Generalization
- 5 Overfitting
- 6 Training ML Models To Avoid Overfitting
- 7 Evaluating the Quality of Classification Models
- 8 Conclusion

Naive Bayes Classifier: An example

How to detect spam e-mails? Input features: Word occurrence in emails (thousands of input features, one per possible word!):

Mail	abacus	...	informatics	...	pills	...	watch	...	zytogenic	Spam
m_1	n	...	y	...	n	...	n	...	n	no
m_2	n	...	n	...	n	...	n	...	n	yes
m_3	n	...	n	...	n	...	n	...	y	no
m_4	n	...	n	...	n	...	n	...	n	yes
m_5	n	...	n	...	n	...	y	...	n	yes
m_6	n	...	n	...	n	...	n	...	n	yes
m_7	n	...	n	...	n	...	n	...	n	no
m_8	n	...	n	...	n	...	n	...	n	yes
m_9	n	...	y	...	n	...	y	...	n	yes

Naive Bayes Classifier: An example

How to detect spam e-mails? Input features: Word occurrence in emails (thousands of input features, one per possible word!):

Mail	abacus	...	informatics	...	pills	...	watch	...	zytogenic	Spam
m_1	n	...	y	...	n	...	n	...	n	no
m_2	n	...	n	...	n	...	n	...	n	yes
m_3	n	...	n	...	n	...	n	...	y	no
m_4	n	...	n	...	n	...	n	...	n	yes
m_5	n	...	n	...	n	...	y	...	n	yes
m_6	n	...	n	...	n	...	n	...	n	yes
m_7	n	...	n	...	n	...	n	...	n	no
m_8	n	...	n	...	n	...	n	...	n	yes
m_9	n	...	y	...	n	...	y	...	n	yes

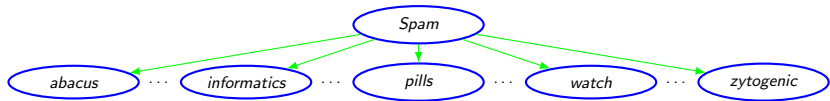
Observation: We can use **Bayesian Networks as ML model**

In this example: Classify email as *spam* if

$$P(\text{Spam} = \text{yes} \mid \mathbf{X} = \mathbf{x}) > \text{threshold}$$

($\mathbf{X} = (\text{abacus}, \dots, \text{zytogenic})$, \mathbf{x} a corresponding set of y/n values)

We will assume the simplest structural assumption:



$$P(a_1, \dots, a_n, \text{Spam}) = P(a_1 \mid \text{Spam}) \cdot P(a_2 \mid \text{Spam}) \cdots P(a_n \mid \text{Spam}) \cdot P(\text{Spam})$$

Making Predictions Using Naive Bayes

Building Naive Bayes as a ML model:

- Training corresponds to setting up the probability tables
- Then we can use inference to perform classification
- **Structural assumption:** Use the simplest possible structure (The target attribute is the cause and the input attributes are all independent symptoms)
 - Simplifies inference
 - We do not need to learn dependencies between variables

Predictions are given by:

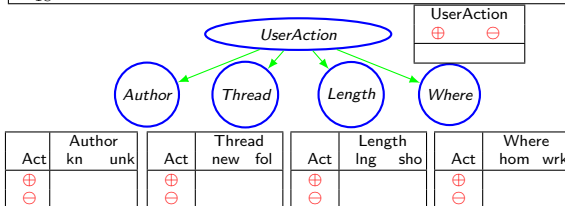
$$P(C=\text{true} \mid a_1 \dots a_n)$$

Learning a Naive Bayes

- Need to learn entries in conditional probability tables. **Use empirical frequencies!**

Probabilities to estimate:

Ex	Author	Thread	Length	WhereRead	UserAction
e ₁	known	new	long	home	skips
e ₂	unknown	new	short	work	reads
e ₃	unknown	follow Up	long	work	skips
e ₄	known	follow Up	long	home	skips
e ₅	known	new	short	home	reads
e ₆	known	follow Up	long	work	skips
e ₇	unknown	follow Up	short	work	skips
e ₈	unknown	new	short	work	reads
e ₉	known	follow Up	long	home	skips
e ₁₀	known	new	long	work	skips
e ₁₁	unknown	follow Up	short	home	skips
e ₁₂	known	new	long	work	skips
e ₁₃	known	follow Up	short	home	reads
e ₁₄	known	new	short	work	reads
e ₁₅	known	new	short	home	reads
e ₁₆	known	follow Up	short	work	reads
e ₁₇	known	new	short	home	reads
e ₁₈	unknown	new	short	work	reads



Learning a Naive Bayes

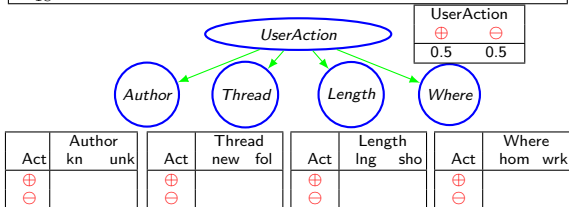
- Need to learn entries in conditional probability tables. **Use empirical frequencies!**

Ex	Author	Thread	Length	WhereRead	UserAction
e ₁	known	new	long	home	skips
e ₂	unknown	new	short	work	reads
e ₃	unknown	follow Up	long	work	skips
e ₄	known	follow Up	long	home	skips
e ₅	known	new	short	home	reads
e ₆	known	follow Up	long	work	skips
e ₇	unknown	follow Up	short	work	skips
e ₈	unknown	new	short	work	reads
e ₉	known	follow Up	long	home	skips
e ₁₀	known	new	long	work	skips
e ₁₁	unknown	follow Up	short	home	skips
e ₁₂	known	new	long	work	skips
e ₁₃	known	follow Up	short	home	reads
e ₁₄	known	new	short	work	reads
e ₁₅	known	new	short	home	reads
e ₁₆	known	follow Up	short	work	reads
e ₁₇	known	new	short	home	reads
e ₁₈	unknown	new	short	work	reads

Probabilities to estimate:

$$P(\text{reads}) = \frac{9}{18}$$

$$P(\text{known}|\text{reads}) =$$



Learning a Naive Bayes

- Need to learn entries in conditional probability tables. **Use empirical frequencies!**

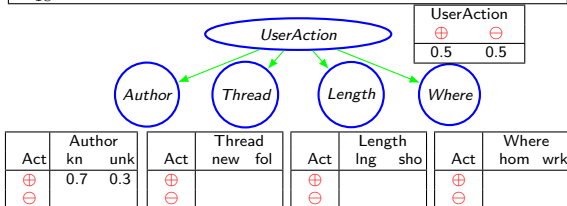
Ex	Author	Thread	Length	WhereRead	UserAction
e ₁	known	new	long	home	skips
e ₂	unknown	new	short	work	reads
e ₃	unknown	follow Up	long	work	skips
e ₄	known	follow Up	long	home	skips
e ₅	known	new	short	home	reads
e ₆	known	follow Up	long	work	skips
e ₇	unknown	follow Up	short	work	skips
e ₈	unknown	new	short	work	reads
e ₉	known	follow Up	long	home	skips
e ₁₀	known	new	long	work	skips
e ₁₁	unknown	follow Up	short	home	skips
e ₁₂	known	new	long	work	skips
e ₁₃	known	follow Up	short	home	reads
e ₁₄	known	new	short	work	reads
e ₁₅	known	new	short	home	reads
e ₁₆	known	follow Up	short	work	reads
e ₁₇	known	new	short	home	reads
e ₁₈	unknown	new	short	work	reads

Probabilities to estimate:

$$P(\text{reads}) = \frac{9}{18}$$

$$P(\text{known}|\text{reads}) = \frac{6}{9}$$

$$P(\text{known}|\text{skips}) =$$



Learning a Naive Bayes

- Need to learn entries in conditional probability tables. **Use empirical frequencies!**

Ex	Author	Thread	Length	WhereRead	UserAction
e ₁	known	new	long	home	skips
e ₂	unknown	new	short	work	reads
e ₃	unknown	follow Up	long	work	skips
e ₄	known	follow Up	long	home	skips
e ₅	known	new	short	home	reads
e ₆	known	follow Up	long	work	skips
e ₇	unknown	follow Up	short	work	skips
e ₈	unknown	new	short	work	reads
e ₉	known	follow Up	long	home	skips
e ₁₀	known	new	long	work	skips
e ₁₁	unknown	follow Up	short	home	skips
e ₁₂	known	new	long	work	skips
e ₁₃	known	follow Up	short	home	reads
e ₁₄	known	new	short	work	reads
e ₁₅	known	new	short	home	reads
e ₁₆	known	follow Up	short	work	reads
e ₁₇	known	new	short	home	reads
e ₁₈	unknown	new	short	work	reads

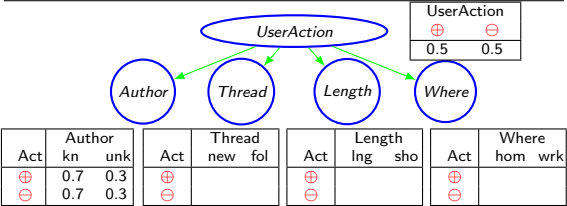
Probabilities to estimate:

$$P(\text{reads}) = \frac{9}{18}$$

$$P(\text{known}|\text{reads}) = \frac{6}{9}$$

$$P(\text{known}|\text{skips}) = \frac{6}{9}$$

$$P(\text{new}|\text{reads}) =$$



Learning a Naive Bayes

- Need to learn entries in conditional probability tables. Use empirical frequencies!

Ex	Author	Thread	Length	WhereRead	UserAction
e ₁	known	new	long	home	skips
e ₂	unknown	new	short	work	reads
e ₃	unknown	follow Up	long	work	skips
e ₄	known	follow Up	long	home	skips
e ₅	known	new	short	home	reads
e ₆	known	follow Up	long	work	skips
e ₇	unknown	follow Up	short	work	skips
e ₈	unknown	new	short	work	reads
e ₉	known	follow Up	long	home	skips
e ₁₀	known	new	long	work	skips
e ₁₁	unknown	follow Up	short	home	skips
e ₁₂	known	new	long	work	skips
e ₁₃	known	follow Up	short	home	reads
e ₁₄	known	new	short	work	reads
e ₁₅	known	new	short	home	reads
e ₁₆	known	follow Up	short	work	reads
e ₁₇	known	new	short	home	reads
e ₁₈	unknown	new	short	work	reads

Probabilities to estimate:

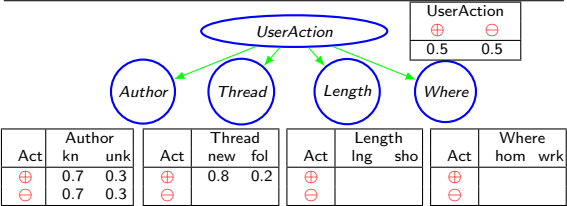
$$P(\text{reads}) = \frac{9}{18}$$

$$P(\text{known}|\text{reads}) = \frac{6}{9}$$

$$P(\text{known}|\text{skips}) = \frac{6}{9}$$

$$P(\text{new}|\text{reads}) = \frac{7}{9}$$

$$P(\text{new}|\text{skips}) =$$



Learning a Naive Bayes

- Need to learn entries in conditional probability tables. **Use empirical frequencies!**

Ex	Author	Thread	Length	WhereRead	UserAction
e ₁	known	new	long	home	skips
e ₂	unknown	new	short	work	reads
e ₃	unknown	follow Up	long	work	skips
e ₄	known	follow Up	long	home	skips
e ₅	known	new	short	home	reads
e ₆	known	follow Up	long	work	skips
e ₇	unknown	follow Up	short	work	skips
e ₈	unknown	new	short	work	reads
e ₉	known	follow Up	long	home	skips
e ₁₀	known	new	long	work	skips
e ₁₁	unknown	follow Up	short	home	skips
e ₁₂	known	new	long	work	skips
e ₁₃	known	follow Up	short	home	reads
e ₁₄	known	new	short	work	reads
e ₁₅	known	new	short	home	reads
e ₁₆	known	follow Up	short	work	reads
e ₁₇	known	new	short	home	reads
e ₁₈	unknown	new	short	work	reads

Probabilities to estimate:

$$P(\text{reads}) = \frac{9}{18}$$

$$P(\text{known}|\text{reads}) = \frac{6}{9}$$

$$P(\text{known}|\text{skips}) = \frac{6}{9}$$

$$P(\text{new}|\text{reads}) = \frac{7}{9}$$

$$P(\text{new}|\text{skips}) = \frac{3}{9}$$

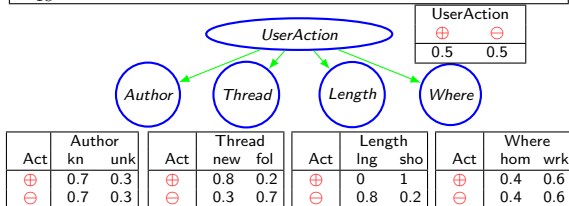
$$P(\text{long}|\text{reads}) = \frac{0}{9}$$

$$P(\text{long}|\text{skips}) = \frac{7}{9}$$

$$P(\text{home}|\text{reads}) = \frac{4}{9}$$

$$P(\text{home}|\text{skips}) = \frac{4}{9}$$

...



Using the Classifier in New Instances

In order to classify a new instance

[Author=unknown, Thread=followUp, Length=short, Where=home]

we calculate

Using the Classifier in New Instances

In order to classify a new instance

[Author=unknown, Thread=followUp, Length=short, Where=home]

we calculate

$$P(\text{skips} \mid \text{unknown, followUp, short, home})$$

Using the Classifier in New Instances

In order to classify a new instance

[Author=unknown, Thread=followUp, Length=short, Where=home]

we calculate

$$P(\text{skips} \mid \text{unknown, followUp, short, home})$$

Using the method from **Chapter 6**, we have

$$P(\text{read})P(\text{unknown}|\text{read})P(\text{followUp}|\text{read})P(\text{short}|\text{read})P(\text{home}|\text{read}) = \frac{9}{18} \cdot \frac{1}{3} \cdot \frac{2}{9} \cdot \frac{9}{9} \cdot \frac{4}{9} = 0.0165;$$

$$P(\text{skips})P(\text{unknown}|\text{skips})P(\text{followUp}|\text{skips})P(\text{short}|\text{skips})P(\text{home}|\text{skips}) = \frac{9}{18} \cdot \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{2}{9} \cdot \frac{4}{9} = 0.0110,$$

which gives

$$P(\text{skips}|\text{unknown, followUp, short, home}) = \frac{0.0110}{0.0110 + 0.0165} = 0.4.$$

Using the Classifier in New Instances

In order to classify a new instance

[Author=unknown, Thread=followUp, Length=short, Where=home]

we calculate

$$P(\text{skips} \mid \text{unknown, followUp, short, home})$$

Using the method from **Chapter 6**, we have

$$P(\text{read})P(\text{unknown}|\text{read})P(\text{followUp}|\text{read})P(\text{short}|\text{read})P(\text{home}|\text{read}) = \frac{9}{18} \cdot \frac{1}{3} \cdot \frac{2}{9} \cdot \frac{9}{9} \cdot \frac{4}{9} = 0.0165;$$

$$P(\text{skips})P(\text{unknown}|\text{skips})P(\text{followUp}|\text{skips})P(\text{short}|\text{skips})P(\text{home}|\text{skips}) = \frac{9}{18} \cdot \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{2}{9} \cdot \frac{4}{9} = 0.0110,$$

which gives

$$P(\text{skips}|\text{unknown, followUp, short, home}) = \frac{0.0110}{0.0110 + 0.0165} = 0.4.$$

However, is this probability reliable?

The Naive Bayes Assumption

Naive Bayes Assumption: All Input attributes are independent given the Target

Example: Given Input Attributes $Cell-1, \dots, Cell-9 \in \{b, w\}$,
predict whether $Symbol=1$.

1	2	3
4	5	6
7	8	9

Question!

Is the Naive Bayes Assumption Realistic?

(A): Never

(B): Almost Never

(C): Almost Always

(D): Always

The Naive Bayes Assumption

Naive Bayes Assumption: All Input attributes are independent given the Target

Example: Given Input Attributes $Cell-1, \dots, Cell-9 \in \{b, w\}$,
predict whether $Symbol=1$.

1	2	3
4	5	6
7	8	9

Question!

Is the Naive Bayes Assumption Realistic?

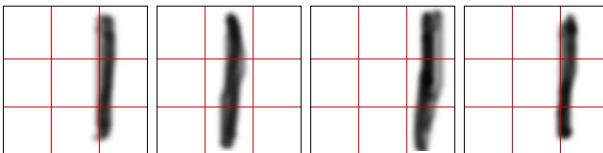
(A): Never

(B): Almost Never

(C): Almost Always

(D): Always

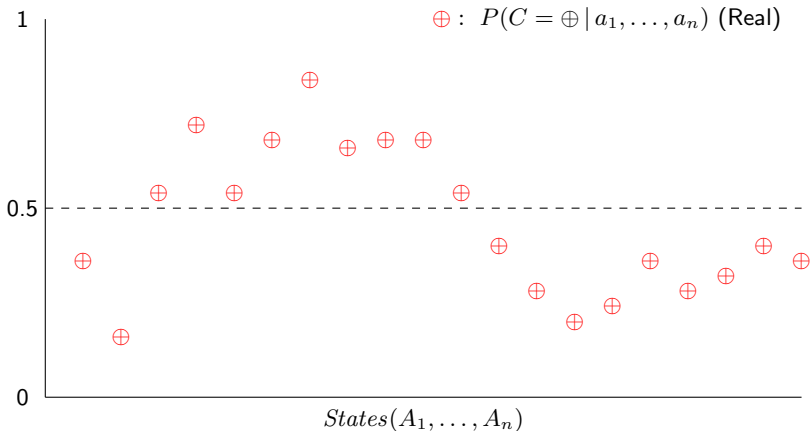
Definitively not in this case, **input attributes not independent given $Symbol=1$!**



~ Naive Bayes assumption often not realistic. Nevertheless, Naive Bayes often successful.

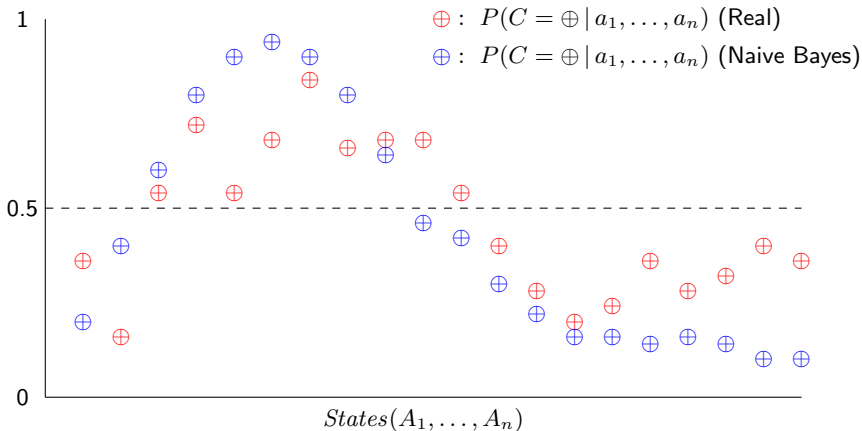
The Paradoxical Success of Naive Bayes

One explanation for the surprisingly good performance of Naive Bayes in many domains: do not require exact distribution for classification, only the right decision boundaries [Domingos, Pazzani 97]



The Paradoxical Success of Naive Bayes

One explanation for the surprisingly good performance of Naive Bayes in many domains: do not require exact distribution for classification, only the right decision boundaries [Domingos, Pazzani 97]



Limitations of Naive Bayes

As happened with Linear Classifiers, no Naive Bayes Classifier can learn a XOR classification:

A	B	Class
yes	yes	\oplus
yes	no	\ominus
no	yes	\ominus
no	no	\oplus

Limitations of Naive Bayes

As happened with Linear Classifiers, no Naive Bayes Classifier can learn a XOR classification:

A	B	Class
yes	yes	\oplus
yes	no	\ominus
no	yes	\ominus
no	no	\oplus

Proof (for reference): Assume it did, then:

- $P(A = y \mid \oplus)P(B = y \mid \oplus)P(\oplus) > P(A = y \mid \ominus)P(B = y \mid \ominus)P(\ominus)$
- $P(A = y \mid \ominus)P(B = n \mid \ominus)P(\ominus) > P(A = y \mid \oplus)P(B = n \mid \oplus)P(\oplus)$
- $P(A = n \mid \ominus)P(B = y \mid \ominus)P(\ominus) > P(A = n \mid \oplus)P(B = y \mid \oplus)P(\oplus)$
- $P(A = n \mid \oplus)P(B = n \mid \oplus)P(\oplus) > P(A = n \mid \ominus)P(B = n \mid \ominus)P(\ominus)$

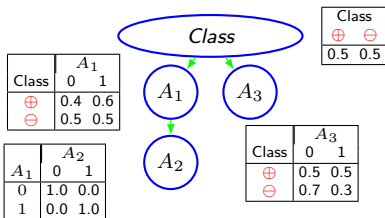
Multiplying the four left sides and the four right sides of these inequalities:

$$\prod_{i=1}^4 (\text{left side of } i.) > \prod_{i=1}^4 (\text{right side of } i.)$$

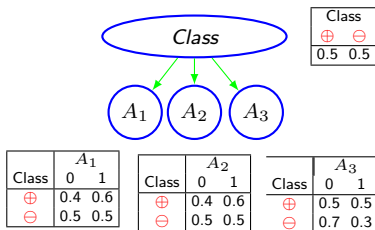
But this is false, because both products are actually equal.

Redundant Input Attributes May Be Harmful

Real representation



Naive Bayes representation



Attribute A_2 is just a duplicate of A_1 .

$$P(\oplus \mid A_1 = 1, A_2 = 1, A_3 = 0) = 0.461$$

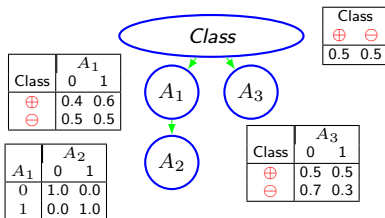
The Naive Bayes model learned from data:

$$P(\oplus \mid A_1 = 1, A_2 = 1, A_3 = 0) = 0.507$$

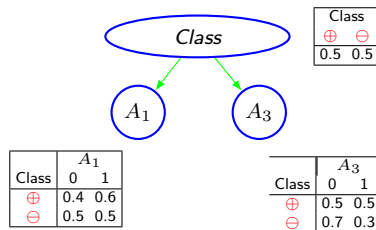
Intuitively: the NB model double counts the information provided by A_1, A_2 . Recall our previous discussion on the use of intermediate variables (c.f., **Chapter 5**)!

Redundant Input Attributes May Be Harmful

Real representation



Naive Bayes representation



Attribute A_2 is just a duplicate of A_1 .

$$P(\oplus \mid A_1 = 1, A_2 = 1, A_3 = 0) = 0.461$$

The Naive Bayes model learned from data:

$$P(\oplus \mid A_1 = 1, A_2 = 1, A_3 = 0) = 0.507$$

Intuitively: the NB model double counts the information provided by A_1, A_2 . Recall our previous discussion on the use of intermediate variables (c.f., **Chapter 5**)!

The Naive Bayes model with selected features A_1 and A_3 :

$$P(\oplus \mid A_1 = 1, A_3 = 0) = 0.461$$

(and all other posterior class probabilities also are the same as in the true model).

Agenda

- 1 Introduction
- 2 Naive Bayes: Using Probabilistic Models
- 3 Case-based Reasoning
- 4 Generalization
- 5 Overfitting
- 6 Training ML Models To Avoid Overfitting
- 7 Evaluating the Quality of Classification Models
- 8 Conclusion

Case-based Reasoning: An intuition

To predict the output feature of a new example e :

- find among the training examples the one (several) most similar to e
- predict the output for e from the known output values of the similar cases

Several names for this approach:

- Instance based learning
- Lazy learning
- Case-based reasoning

Required: **distance measure** on values of input features.

Distance Measures

Distances for numeric features

If all features \mathbf{X} are numeric:

- Euclidean distance: $d(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_i (x_i - x'_i)^2}$
- Manhattan distance: $d(\mathbf{x}, \mathbf{x}') = \sum_i |x_i - x'_i|$

Distances for discrete features

For a single feature X with domain $\{x_1, \dots, x_k\}$:

- Zero-One distance: $d(x_i, x_j) = 0$ if $j = i$, $d(x_i, x_j) = 1$ if $j \neq i$
- Distance matrix: specify for each pair x_i, x_j the distance $d(x_i, x_j)$ in a $k \times k$ -matrix. Example:

	<i>low</i>	<i>medium</i>	<i>high</i>
<i>low</i>	0	2	5
<i>medium</i>	2	0	1
<i>high</i>	5	1	0

For a set of discrete features \mathbf{X} :

- Define distance d_i and weight w_i for each $X_i \in \mathbf{X}$
- Define $d(\mathbf{x}, \mathbf{x}') = \sum_i w_i d_i(x_i, x'_i)$

K -Nearest-Neighbor Classifier

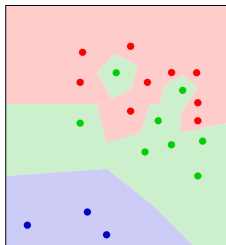
Given

- training examples (\mathbf{x}_i, y_i) ($i = 1, \dots, N$)
- a new case \mathbf{x} to be classified
- a distance measure $d(\mathbf{x}, \mathbf{x}')$

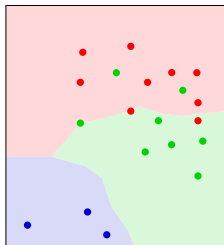
classify \mathbf{x} as follows:

- find the K training examples $\mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_K}$ with minimal distance to \mathbf{x}
- predict for \mathbf{x} the most frequent value in y_{i_1}, \dots, y_{i_K} .

Example



1-Nearest Neighbor

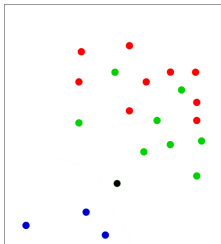


5-Nearest Neighbor

- Output feature:
red, blue, green
- Two numeric input features
- Euclidean distance
- Colored dots: training examples
- Colored regions: regions of input values that will be classified as that color

Questionnaire!

If we had a new input, represented as the black point.



Question!

What will be the prediction for the black point when applying 3-Nearest Neighbor?

(A): Red

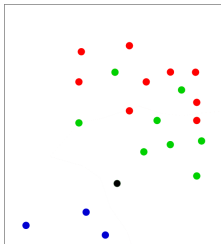
(B): Blue

(C): Green

(D): Undefined

Questionnaire!

If we had a new input, represented as the black point.



Question!

What will be the prediction for the black point when applying 3-Nearest Neighbor?

(A): Red

(B): Blue

(C): Green

(D): Undefined

Blue. We pick the three closest points to the black one. They are two blue and one green, so the predicted color is blue.

Agenda

- 1 Introduction
- 2 Naive Bayes: Using Probabilistic Models
- 3 Case-based Reasoning
- 4 Generalization**
- 5 Overfitting
- 6 Training ML Models To Avoid Overfitting
- 7 Evaluating the Quality of Classification Models
- 8 Conclusion

What is to Learn?

Reminder: Learning Task (**Chapter 7**)

Find the model in the hypothesis space that minimizes the error on the training data.

What is to Learn?

Reminder: Learning Task (**Chapter 7**)

Find the model in the hypothesis space that minimizes the error on the training data.

This is not true!

What is to Learn?

Reminder: Learning Task (Chapter 7)

Find the model in the hypothesis space that minimizes the error on the training data.

This is not true!

Technically, we already know the answer on the training set. What we care about is that the model gives correct predictions on new examples!

Is that even possible? We do not know what the future examples will be!

Question!

What is the next number in the sequence? 1, 3, 5, 7, ?

(A): 9

(B): 11

(C): 13

(D): 217341

What is to Learn?

Find the next number of the sequence

1, 3, 5, 7, ?

Correct solution

217341

because when

$$f(x) = \frac{18111}{2}x^4 - 90555x^3 + \frac{633885}{2}x^2 - 452773x + 217331$$

$$f(1)=1$$

$$f(2)=3$$

$$f(3)=5$$

$$f(4)=7$$

$$f(5)=217341$$

much solution

wow very logic

Such solution



What is to Learn?

Reminder: Learning Task (Chapter 7)

Find the model in the hypothesis space that minimizes the error on the training data.

This is not true!

Technically, we already know the answer on the training set. What we care about is that the model gives correct predictions on new examples!

Is that even possible? We do not know what the future examples will be!

Question!

What is the next number in the sequence? 1, 3, 5, 7, ?

(A): 9

(B): 11

(C): 13

(D): 217341

For any constant c , we can find a function such that $f(1) = 1$, $f(2) = 3$, $f(3) = 5$, $f(4) = 7$, and $f(5) = c$.

However, the simplest function is $f(x) = 1 + 2x$, so the most likely number in the sequence is 9.

→ **Occam's Razor Principle:** Simpler models are to be preferred

Generalization and The Big Assumption Behind Machine Learning

Assumption

The Training data and the Future data are taken from the same probability distribution

In other words: **training examples are representative of future examples.**

Generalization and The Big Assumption Behind Machine Learning

Assumption

The Training data and the Future data are taken from the same probability distribution

In other words: **training examples are representative of future examples.**

When collecting data, we need to make sure that this holds (as much as possible):

- If you predict the height and only train on children from 5 to 14 years old, what will the model say about a 60-year old person?

Generalization and The Big Assumption Behind Machine Learning

Assumption

The Training data and the Future data are taken from the same probability distribution

In other words: **training examples are representative of future examples.**

When collecting data, we need to make sure that this holds (as much as possible):

- If you predict the height and only train on children from 5 to 14 years old, what will the model say about a 60-year old person? → More than 3 meters!
- If your face recognition dataset has mostly white people, you can make the news:
<https://www.wired.com/story/best-algorithms-struggle-recognize-black-faces-equally/>

Agenda

- 1 Introduction
- 2 Naive Bayes: Using Probabilistic Models
- 3 Case-based Reasoning
- 4 Generalization
- 5 Overfitting**
- 6 Training ML Models To Avoid Overfitting
- 7 Evaluating the Quality of Classification Models
- 8 Conclusion

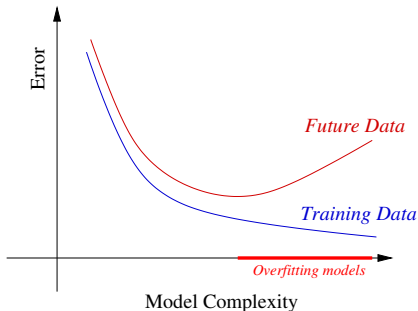
Overfitting

Noise in data may lead to a bad classifier. In particular, if the decision tree fits the data perfectly. This is called **overfitting**.

Definition

A hypothesis h is said to overfit the training data if there exists some alternative hypotheses h' , such that:

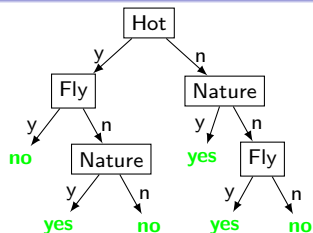
- h has smaller error than h' over the training data, but
- h' has a smaller error than h over the entire distribution of instances.



Overfitting: Decision Trees

Decision tree learned from the holiday data on the left

Culture	Fly	Hot	Music	Nature	Likes
no	no	yes	no	no	no
no	yes	yes	no	no	no
yes	yes	yes	yes	yes	no
no	yes	yes	yes	yes	no
yes	no	no	yes	yes	yes
no	no	no	no	no	no
no	no	no	yes	yes	yes
yes	yes	yes	no	no	no
yes	yes	no	yes	yes	yes
yes	yes	no	no	no	yes
yes	no	yes	no	yes	yes
no	no	no	yes	no	no
yes	no	yes	yes	no	no
yes	yes	yes	yes	no	no
yes	no	no	yes	no	no
yes	yes	yes	no	yes	no
no	no	no	no	yes	yes
no	yes	no	no	no	yes



Trees provide very accurate representation of training examples, but are they the best trees to predict the preferences of *future* customers?

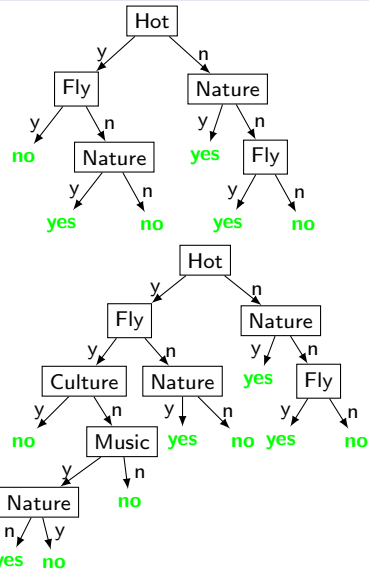
Overfitting: Decision Trees

Decision tree learned from the holiday data on the left

Culture	Fly	Hot	Music	Nature	Likes
no	no	yes	no	no	no
no	yes	yes	no	no	no
yes	yes	yes	yes	yes	no
no	yes	yes	yes	yes	no
no	yes	yes	no	yes	no
yes	no	no	yes	yes	yes
no	no	no	no	no	no
no	no	no	yes	yes	yes
yes	yes	yes	no	no	no
yes	yes	no	yes	yes	yes
yes	yes	no	no	no	yes
yes	no	yes	no	yes	yes
no	no	no	yes	no	no
yes	no	yes	yes	no	no
yes	yes	yes	yes	no	no
yes	no	no	yes	no	no
yes	yes	yes	no	yes	no
no	no	no	no	yes	yes
no	yes	no	no	no	yes

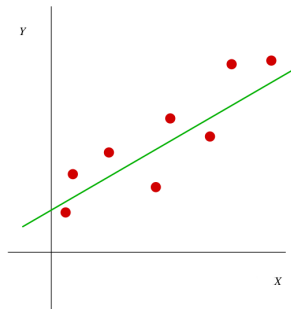
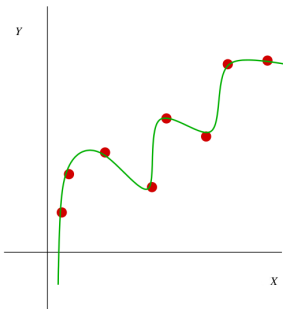
What happen if we add one more example?

Culture	Fly	Hot	Music	Nature	Likes
no	yes	yes	yes	no	yes



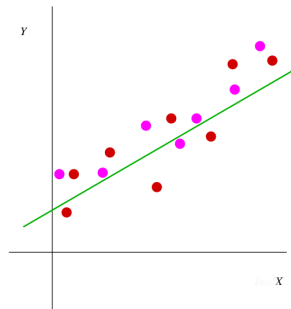
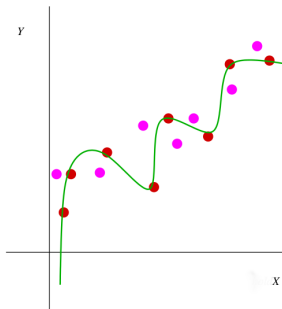
Trees provide very accurate representation of training examples, but are they the best trees to predict the preferences of *future* customers?

Overfitting: Neural Networks



Left model: minimizes SSE on training examples

Overfitting: Neural Networks



Left model: minimizes SSE on training examples

Right model: may have smaller SSE on **future observations**

Overfitting: Naive Bayes

When learning the naive Bayes model we estimated $P(\text{long}|\text{reads})$ as

$$P(\text{long}|\text{reads}) = \frac{0}{9},$$

based on 9 cases only \leadsto unreliable parameter estimates and risk of zero probabilities.

Overfitting: Naive Bayes

When learning the naive Bayes model we estimated $P(\text{long}|\text{reads})$ as

$$P(\text{long}|\text{reads}) = \frac{0}{9},$$

based on 9 cases only \leadsto unreliable parameter estimates and risk of zero probabilities.

Solution: using pseudo counts

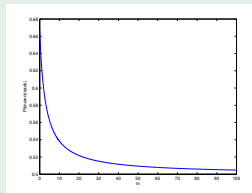
$$P(A = a|C = c) = \frac{N(A = a, C = c) + p_{ac} \cdot m}{N(C = c) + m}$$

where

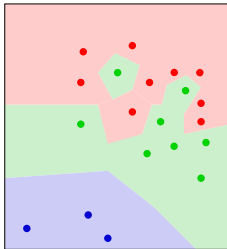
- p_{ac} is our prior estimate of the probability (often chosen as a uniform distribution) and
- m is a virtual sample size (determining the weight of p_{ac} relative to the observed data).

Example

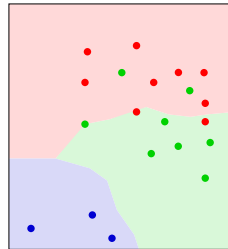
$$P(\text{known}|\text{reads}) = \frac{2 + 0.5 \cdot m}{3 + m}$$



Overfitting: Nearest Neighbor



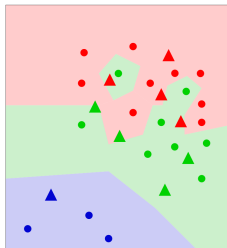
1-Nearest Neighbor



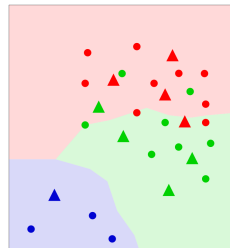
5-Nearest Neighbor

1-Nearest Neighbor correctly classifies all training examples

Overfitting: Nearest Neighbor



1-Nearest Neighbor

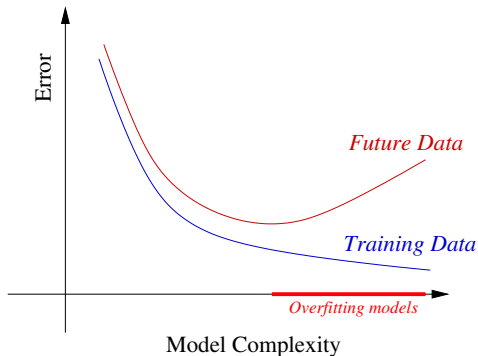


5-Nearest Neighbor

1-Nearest Neighbor correctly classifies all training examples

5-Nearest Neighbor may be better for future observations (triangles)

The General Picture



Model	Complexity	Error
Decision Tree	Depth, # Nodes	Classification error
Neural Network	# hidden nodes/layers	SSE
Nearest Neighbor	Decision regions	Classification error

A model **overfits** the training data, if a smaller error on future data would be obtained by a less complex model.

Agenda

- 1 Introduction
- 2 Naive Bayes: Using Probabilistic Models
- 3 Case-based Reasoning
- 4 Generalization
- 5 Overfitting
- 6 Training ML Models To Avoid Overfitting**
- 7 Evaluating the Quality of Classification Models
- 8 Conclusion

Avoiding Overfitting

In this section, we see three separate ideas to avoid overfitting:

- 1 Keep the models simple
- 2 Evaluate our models to check if they are overfitting
- 3 Smooth Training Data (Using Pseudo-counts in Naive Bayes, see slide 27)

Avoiding Overfitting by Preferring Simpler Models

Reduced hypothesis space

Do not allow overly complex models by setting up hyperparameters:

- Naive Bayes model (instead of more complex Bayesian models)
- K -NN for “large” K
- Maximum tree size in decision trees

Avoiding Overfitting by Preferring Simpler Models

Reduced hypothesis space

Do not allow overly complex models by setting up hyperparameters:

- Naive Bayes model (instead of more complex Bayesian models)
- K -NN for “large” K
- Maximum tree size in decision trees

Modified Search/Scoring (Regularization)

Do not allow to learn models that are too complex (relative to the available data):

- Decision Trees: use an early stopping criterion, e.g. stop construction when (sub-) set of training examples contains fewer than k elements (for not too small k).

Avoiding Overfitting by Preferring Simpler Models

Reduced hypothesis space

Do not allow overly complex models by setting up hyperparameters:

- Naive Bayes model (instead of more complex Bayesian models)
- K -NN for “large” K
- Maximum tree size in decision trees

Modified Search/Scoring (Regularization)

Do not allow to learn models that are too complex (relative to the available data):

- Decision Trees: use an early stopping criterion, e.g. stop construction when (sub-) set of training examples contains fewer than k elements (for not too small k).
- Add to evaluation measure a **penalty term** for complexity. This penalty term can have an interpretation as a **prior model probability**, or **model description length**.

Example:

$$\text{minimize } \underbrace{\sum_{e \in E} (Y(e) - f(e))^2 / |E|}_{\text{MSE error}} + \underbrace{\text{complexity}(f)}_{\text{Regularization term}}$$

→ These techniques will usually lead to more complex models only when the data strongly supports it (especially: large number of examples).

Evaluate our Models to Check if they are Overfitting

Reserve part of the training examples as a **validation set**:

- not used during training
- used as proxy for future data in model evaluation

Evaluate our Models to Check if they are Overfitting

Reserve part of the training examples as a **validation set**:

- not used during training
- used as proxy for future data in model evaluation

Simplest approach: split the available data randomly into a **training** and a **validation** set. Typically: 50%/50% or 66%/33% split.

(Note: in some cases a random split is not good. For example, in time-series is better to use the latest data as validation/test set)

Evaluate our Models to Check if they are Overfitting

Reserve part of the training examples as a **validation set**:

- not used during training
- used as proxy for future data in model evaluation

Simplest approach: split the available data randomly into a **training** and a **validation** set. Typically: 50%/50% or 66%/33% split.

(Note: in some cases a random split is not good. For example, in time-series is better to use the latest data as validation/test set)

Observation: If the validation test is used for tuning the learning algorithm is not really anymore “future” data, so we need yet more data to determine the quality of our final model!

Evaluate our Models to Check if they are Overfitting

Reserve part of the training examples as a **validation set**:

- not used during training
- used as proxy for future data in model evaluation

Simplest approach: split the available data randomly into a **training** and a **validation** set. Typically: 50%/50% or 66%/33% split.

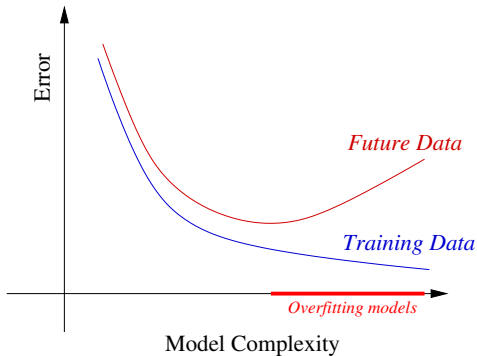
(Note: in some cases a random split is not good. For example, in time-series is better to use the latest data as validation/test set)

Observation: If the validation test is used for tuning the learning algorithm is not really anymore “future” data, so we need yet more data to determine the quality of our final model!

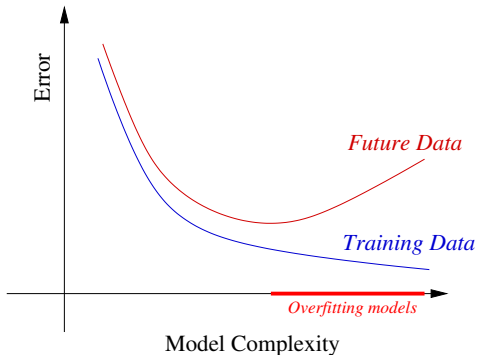
We divide our data into three sub-sets:

- **Training set**: Used for the learning algorithms
- **Validation set**: Used for tuning hyperparameters and/or decide when to stop the training
- **Test set**: Used for evaluating how good the resulting model is (e.g. to report accuracy on the test set)

Using Validation Test as Stopping Condition



Using Validation Test as Stopping Condition



- While training we measure error on validation set
- We stop training if error starts increasing

Using Validation Data on Decision Trees

Post pruning

Use of validation set in decision tree learning:

- Build decision tree using *training* set
- For each internal node:
 - test whether accuracy on *validation* set is improved by replacing sub-tree rooted at this node by single leaf (labeled with most frequent target feature value of *training* examples in this sub-tree)
 - if yes: replace sub-tree with leaf (*prune* sub-tree)

Tuning Hyperparameters

Selection of K in K -NN

for $K = 1, 2, 3, \dots$:

- measure accuracy of K -NN based on *training* examples for *validation* examples

Next:

- use K with best performance on *validation* examples
- *validation* examples can now be merged with *training* examples for predicting future cases

Tuning Hyperparameters

Selection of K in K -NN

for $K = 1, 2, 3, \dots$:

- measure accuracy of K -NN based on *training* examples for *validation* examples

Next:

- use K with best performance on *validation* examples
- *validation* examples can now be merged with *training* examples for predicting future cases

Selection of Neural Network Structure

for $\#hl = 1, 2, \dots, \max$, and $\#nd = 1, 2, \dots, \max$:

- learn Neural Network with $\#hl$ hidden layers and $\#nd$ nodes per hidden layer using *training* examples
- evaluate SSE of learned network on *validation* examples

Next:

- select network structure with minimal SSE
- re-learn weights using merged training and validation examples

Cross-Validation

Disadvantage of training/validation splits (for small datasets):

- the examples in the validation set are partly “wasted”
- (unrepresentative) patterns in the validation set can bias the model selection

Cross-Validation

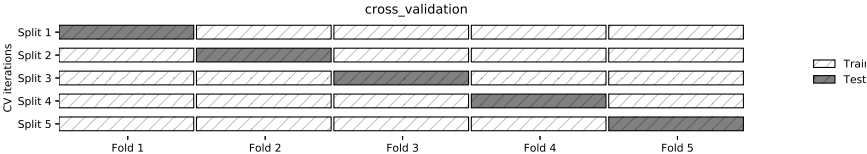
Disadvantage of training/validation splits (for small datasets):

- the examples in the validation set are partly “wasted”
- (unrepresentative) patterns in the validation set can bias the model selection

Cross Validation

The *n*-fold cross-validation approach:

- Divide the examples into *n* equal sized subsets, or *folds* (e.g. $n = 10$)
- for $i = 1, \dots, n$:
 - learn model (with given “complexity setting”) using folds $1, \dots, i - 1, i + 1, \dots, n$
 - evaluate using fold *i*
 - average the evaluation scores from the *n* folds
- Choose “complexity setting” that obtained highest average evaluation score
- Learn final model with chosen “complexity setting” using all available examples



Agenda

- 1 Introduction
- 2 Naive Bayes: Using Probabilistic Models
- 3 Case-based Reasoning
- 4 Generalization
- 5 Overfitting
- 6 Training ML Models To Avoid Overfitting
- 7 Evaluating the Quality of Classification Models**
- 8 Conclusion

Evaluating the Quality of Classification Models

Accuracy: Measure the percentage of **correct** classifications.

Evaluating the Quality of Classification Models

Accuracy: Measure the percentage of **correct classifications**.

Learned Model A

	Predicted		
	NO	YES	
Actual: NO	5	10	15
Actual: YES	8	142	150
	13	152	165

Learned Model B

	Predicted		
	NO	YES	
Actual: NO	0	15	15
Actual: YES	0	150	150
	0	165	165

Learned Model C

	Predicted		
	NO	YES	
Actual: NO	15	0	15
Actual: YES	20	130	150
	35	130	165

Question!

Which of these classifiers is better?

Evaluating the Quality of Classification Models

Accuracy: Measure the percentage of **correct classifications**.

Learned Model A

	Predicted		
	NO	YES	
Actual: NO	5	10	15
Actual: YES	8	142	150
	13	152	165

Learned Model B

	Predicted		
	NO	YES	
Actual: NO	0	15	15
Actual: YES	0	150	150
	0	165	165

Learned Model C

	Predicted		
	NO	YES	
Actual: NO	15	0	15
Actual: YES	20	130	150
	35	130	165

Question!

Which of these classifiers is better?

We get the following accuracy for each of the models:
Accuracy of A

Evaluating the Quality of Classification Models

Accuracy: Measure the percentage of **correct classifications**.

Learned Model A

	Predicted		
	NO	YES	
Actual: NO	5	10	15
Actual: YES	8	142	150
	13	152	165

Learned Model B

	Predicted		
	NO	YES	
Actual: NO	0	15	15
Actual: YES	0	150	150
	0	165	165

Learned Model C

	Predicted		
	NO	YES	
Actual: NO	15	0	15
Actual: YES	20	130	150
	35	130	165

Question!

Which of these classifiers is better?

We get the following accuracy for each of the models:

Accuracy of A $\frac{147}{165} \simeq 89\%$

Accuracy of B $\frac{150}{165} \simeq 91\%$

Accuracy of C $\frac{145}{165} \simeq 88\%$

Evaluating the Quality of Classification Models

Accuracy: Measure the percentage of **correct classifications**.

Learned Model A

	Predicted		
	NO	YES	
Actual: NO	5	10	15
Actual: YES	8	142	150
	13	152	165

Learned Model B

	Predicted		
	NO	YES	
Actual: NO	0	15	15
Actual: YES	0	150	150
	0	165	165

Learned Model C

	Predicted		
	NO	YES	
Actual: NO	15	0	15
Actual: YES	20	130	150
	35	130	165

Question!

Which of these classifiers is better?

We get the following accuracy for each of the models:

Accuracy of A $\frac{147}{165} \simeq 89\%$

Accuracy of B $\frac{150}{165} \simeq 91\%$

Accuracy of C $\frac{145}{165} \simeq 88\%$

- *B* has higher accuracy but always predicting YES is not that useful.
- Between *A* and *C* it depends on the consequences of having a mistake on NO or YES cases

→ **Accuracy does not tell the full story, specially on unbalanced datasets**

Confusion Matrix

Four possible outcomes for each test example:

- **TRUE Positive** (TP): the model correctly predicts the positive class
- **TRUE Negative** (TN): the model correctly predicts the negative class
- **FALSE Positive** (FP): the model incorrectly predicts the positive class
- **FALSE Negative** (FN): the model incorrectly predicts the negative class

Confusion Matrix

Four possible outcomes for each test example:

- **TRUE Positive** (TP): the model correctly predicts the positive class
- **TRUE Negative** (TN): the model correctly predicts the negative class
- **FALSE Positive** (FP): the model incorrectly predicts the positive class
- **FALSE Negative** (FN): the model incorrectly predicts the negative class

	Predicted: NO	Predicted: YES	
Actual: NO	TN	FP	
Actual: YES	FN	TP	
			n

Therefore **Accuracy** is given by $\frac{TP+TN}{\text{Total}}$

Confusion Matrix

Four possible outcomes for each test example:

- **TRUE Positive** (TP): the model correctly predicts the positive class
- **TRUE Negative** (TN): the model correctly predicts the negative class
- **FALSE Positive** (FP): the model incorrectly predicts the positive class
- **FALSE Negative** (FN): the model incorrectly predicts the negative class

	Predicted: NO	Predicted: YES	
Actual: NO	TN	FP	
Actual: YES	FN	TP	
			n

Therefore **Accuracy** is given by $\frac{TP+TN}{\text{Total}}$

→ But we can also compute other metrics: precision, recall, specificity

Precision

Measure the **estimated performance** when TN cannot be assessed.

Precision is given by $\frac{TP}{\text{Predicted YES}} = \frac{TP}{TP+FP}$

Precision

Measure the **estimated performance** when TN cannot be assessed.

Precision is given by $\frac{TP}{\text{Predicted YES}} = \frac{TP}{TP+FP}$

- An issue with accuracy (and specificity) is that TN cannot always be counted. (Example: TN of a Google search query?)
- In those cases, use precision instead: When the model predicts YES, how often is it correct?
- Precision is the estimated probability that a randomly selected retrieved document is relevant.

Precision

Measure the **estimated performance** when TN cannot be assessed.

Precision is given by $\frac{TP}{\text{Predicted YES}} = \frac{TP}{TP+FP}$

- An issue with accuracy (and specificity) is that TN cannot always be counted. (Example: TN of a Google search query?)
- In those cases, use precision instead: When the model predicts YES, how often is it correct?
- Precision is the estimated probability that a randomly selected retrieved document is relevant.

	Predicted: NO	Predicted: YES	
Actual: NO	$TN = 50$	$FP = 10$	60
Actual: YES	$FN = 5$	$TP = 100$	105
	55	110	n=165

Precision is given by

Precision

Measure the **estimated performance** when TN cannot be assessed.

Precision is given by $\frac{TP}{\text{Predicted YES}} = \frac{TP}{TP+FP}$

- An issue with accuracy (and specificity) is that TN cannot always be counted. (Example: TN of a Google search query?)
- In those cases, use precision instead: When the model predicts YES, how often is it correct?
- Precision is the estimated probability that a randomly selected retrieved document is relevant.

	Predicted: NO	Predicted: YES	
Actual: NO	$TN = 50$	$FP = 10$	60
Actual: YES	$FN = 5$	$TP = 100$	105
	55	110	n=165

Precision is given by $\frac{TP}{TP+FP} = \frac{100}{110} = 91\%$

Recall (True-Positive Rate or Sensitivity)

Measure the performance on YES instances. It measures the **sensitivity** of the model.

Recall is given by $\frac{TP}{\text{Actual YES}} = \frac{TP}{TP+FN}$

Recall (True-Positive Rate or Sensitivity)

Measure the performance on YES instances. It measures the **sensitivity** of the model.

Recall is given by $\frac{TP}{\text{Actual YES}} = \frac{TP}{TP+FN}$

- When the true value is YES, how often does the classifier predict YES?
- It is specially important in some applications (like medical applications), where the objective is to detect all positive cases, even if accuracy or precision is affected.
- Other example, recall is the estimated probability that a randomly selected relevant document is retrieved in a search

Recall (True-Positive Rate or Sensitivity)

Measure the performance on YES instances. It measures the **sensitivity** of the model.

Recall is given by $\frac{TP}{\text{Actual YES}} = \frac{TP}{TP+FN}$

- When the true value is YES, how often does the classifier predict YES?
- It is specially important in some applications (like medical applications), where the objective is to detect all positive cases, even if accuracy or precision is affected.
- Other example, recall is the estimated probability that a randomly selected relevant document is retrieved in a search

	Predicted: NO	Predicted: YES	
Actual: NO	$TN = 50$	$FP = 10$	60
Actual: YES	$FN = 5$	$TP = 100$	105
	55	110	n=165

Recall is given by

Recall (True-Positive Rate or Sensitivity)

Measure the performance on YES instances. It measures the **sensitivity** of the model.

Recall is given by $\frac{TP}{\text{Actual YES}} = \frac{TP}{TP+FN}$

- When the true value is YES, how often does the classifier predict YES?
- It is specially important in some applications (like medical applications), where the objective is to detect all positive cases, even if accuracy or precision is affected.
- Other example, recall is the estimated probability that a randomly selected relevant document is retrieved in a search

	Predicted: NO	Predicted: YES	
Actual: NO	$TN = 50$	$FP = 10$	60
Actual: YES	$FN = 5$	$TP = 100$	105
	55	110	n=165

Recall is given by $\frac{TP}{TP+FN} = \frac{100}{105} = 95\%$

Specificity

Measure the performance on NO instances.

Specificity is given by $\frac{TN}{\text{Actual NO}} = \frac{TN}{TN + FP}$

Specificity

Measure the performance on NO instances.

Specificity is given by $\frac{TN}{\text{Actual NO}} = \frac{TN}{TN + FP}$

- When the true value is NO, how often does the classifier predict NO?
- True negative rate: proportion of actual negatives that the classifier correctly identifies.

Specificity

Measure the performance on NO instances.

Specificity is given by $\frac{TN}{\text{Actual NO}} = \frac{TN}{TN+FP}$

- When the true value is NO, how often does the classifier predict NO?
- True negative rate: proportion of actual negatives that the classifier correctly identifies.

	Predicted: NO	Predicted: YES	
Actual: NO	$TN = 50$	$FP = 10$	60
Actual: YES	$FN = 5$	$TP = 100$	105
	55	110	n=165

Specificity is given by

Specificity

Measure the performance on NO instances.

Specificity is given by $\frac{TN}{\text{Actual NO}} = \frac{TN}{TN+FP}$

- When the true value is NO, how often does the classifier predict NO?
- True negative rate: proportion of actual negatives that the classifier correctly identifies.

	Predicted: NO	Predicted: YES	
Actual: NO	$TN = 50$	$FP = 10$	60
Actual: YES	$FN = 5$	$TP = 100$	105
	55	110	n=165

Specificity is given by $\frac{TN}{TN+FP} = \frac{50}{60} = 83\%$

Probabilistic Interpretation of the Evaluation Metrics

- **Precision** is the probability that a random patient from all those with a positive test indeed has cancer:

$$\frac{TP}{\text{Predicted YES}} = \frac{P(\text{Cancer} = \text{YES}, \text{Classifier} = \text{YES})}{P(\text{Classifier} = \text{YES})}$$

$$P(\text{Real} = \text{YES} | \text{Prediction} = \text{YES})$$

- **Recall** is the probability of a positive test given that the patient has cancer:

$$\frac{TP}{\text{Actual YES}} = \frac{P(\text{Classifier} = \text{YES}, \text{Cancer} = \text{YES})}{P(\text{Cancer} = \text{YES})}$$

$$P(\text{Prediction} = \text{YES} | \text{Real} = \text{YES})$$

- **Specificity** is the probability of a negative test given that the patient does not have cancer:

$$\frac{TN}{\text{Actual NO}} = \frac{P(\text{Classifier} = \text{NO}, \text{Cancer} = \text{NO})}{P(\text{Cancer} = \text{NO})}$$

$$P(\text{Prediction} = \text{NO} | \text{Real} = \text{NO})$$

F1 Score

- We must consider pairs of measures such as recall/ specificity or recall/ precision for interpreting the performance of a classifier
- **Observation:** Models that have 0 precision or 0 recall are worthless (they simply classify every instance as positive or negative)
- Precision and Recall are two sides of the same coin, can we summarize them in a single metric?

The **F1 score** provides a single measure that summarizes the overall performance:

$$F_1 = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

- F1 is the harmonic mean between precision and recall
- Maximum value of F1 is 1 (perfect precision and recall)
- Minimum value of F1 is 0 (when recall or precision are 0)
- F1 by design does not take TN into account!

Confusion Matrix with more classes

<div> <div>Predicted fruit</div> <div>Actual fruit</div> </div>	Perceived			
	Apple	Banana	Orange	Clementine
Apple	10	1	2	2
Banana	5	18	1	2
Orange	8	3	15	11
Clementine	5	3	8	12

→ We need to compute precision and recall for each individual class!

Agenda

- 1 Introduction
- 2 Naive Bayes: Using Probabilistic Models
- 3 Case-based Reasoning
- 4 Generalization
- 5 Overfitting
- 6 Training ML Models To Avoid Overfitting
- 7 Evaluating the Quality of Classification Models
- 8 Conclusion**

Summary

- **Naive Bayes** is a simple way of using probabilistic reasoning to perform classification. Despite making strong assumptions it has good accuracy on many applications.
- **Case-based reasoning** methods perform classification without an expensive training phase
- **Overfitting** is a common problem when the learned model minimizes the error on the training set at expenses of having more error on future examples
- We can address overfitting by keeping our hypothesis space simple, or by using a validation dataset
- There are many metrics to measure the performance of a classifier, such as **accuracy**, **precision**, and **recall**.

Reading

- *Chapter 7.7 Case-Based Reasoning* from the book "Artificial Intelligence: Foundations of Computational Agents (2nd edition)
- *Chapter 7.4 Overfitting* from the book "Artificial Intelligence: Foundations of Computational Agents (2nd edition)
- *Chapter 10.1 Probabilistic learning* (only 10.1.1 and 10.1.2) from the book "Artificial Intelligence: Foundations of Computational Agents (2nd edition)
- Extra Reading: To go much further, you can read the Lecture Notes of the Stanford Course in Machine Learning. In this lecture we cover