Machine Intelligence

Lecture 9: Learning

Thomas Dyhre Nielsen

Aalborg University

Tentative course overview

Topics:

- Introduction
- Search-based methods
- Constrained satisfaction problems
- Logic-based knowledge representation
- Representing domains endowed with uncertainty.
- Bayesian networks
- Inference in Bayesian networks
- **Machine learning**
- Planning
- Clustering
- Multi-agent systems

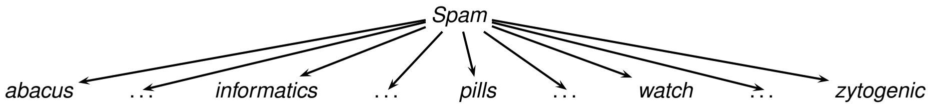
Probabilistic Models

Spam Data

Word occurrence in emails (thousands of input features!):

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



Classify email as *spam* if

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

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

Structural assumption

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

Learning

- Need to learn entries in conditional probability tables.
- Simplest approach: use **empirical frequencies**, e.g:

$$P(\textit{pills} = y \mid \textit{Spam} = \textit{yes}) = \frac{\#\text{mails with } \textit{pills} = y \text{ and } \textit{Spam} = \textit{yes}}{\#\text{mails with } \textit{Spam} = \textit{yes}}$$

Example: Learning a Naive Bayes

Example	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:

NB model:

Example: Learning a Naive Bayes

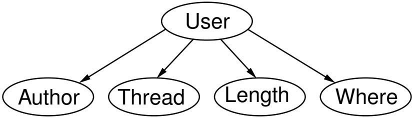
Example	Author	Thread	Length	WhereRead	UserAction
e_1	known	new	long	home	skips
e_2	unknown	new	short	work	reads
e_3	unknown	follow Up	long	work	skips
e_4	known	follow Up	long	home	skips
e_5	known	new	short	home	reads
e_6	known	follow Up	long	work	skips
e_7	unknown	follow Up	short	work	skips
e_8	unknown	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:

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

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

NB model:



Example: Learning a Naive Bayes

Example	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

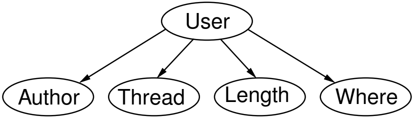
Probabilities to estimate:

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

$$P(\text{known}|\text{reads}) = \frac{2}{3}$$

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

NB model:



Example: Learning a Naive Bayes

Example	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:

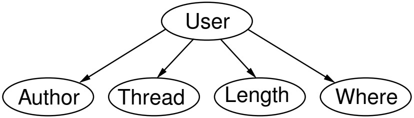
$$P(\text{reads}) = \frac{9}{18}$$

$$P(\text{known}|\text{reads}) = \frac{2}{3}$$

$$P(\text{known}|\text{skips}) = \frac{2}{3}$$

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

NB model:



Example: Learning a Naive Bayes

Example	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:

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

$$P(\text{known}|\text{reads}) = \frac{2}{3}$$

$$P(\text{known}|\text{skips}) = \frac{2}{3}$$

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

$$P(\text{new}|\text{skips}) = \frac{1}{3}$$

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

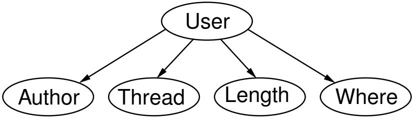
$$P(\text{long}|\text{skips}) = \frac{7}{9}$$

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

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

$$\ldots$$

NB model:



Making predictions using Naive Bayes

In order to classify a new instance

$$[\text{ Author=unknown, Thread=followUp, Length=short, Where=home}]$$

we calculate

Making predictions using Naive Bayes

In order to classify a new instance

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

we calculate

$$P(\text{skips}|\text{unknown}, \text{followUp}, \text{short}, \text{home})$$
$$= \frac{P(\text{skips}, \text{unknown}, \text{followUp}, \text{short}, \text{home})}{P(\text{skips}, \text{unknown}, \text{followUp}, \text{short}, \text{home}) + P(\text{reads}, \text{unknown}, \text{followUp}, \text{short}, \text{home})}$$

Making predictions using Naive Bayes

In order to classify a new instance

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

we calculate

$$P(\text{skips}|\text{unknown}, \text{followUp}, \text{short}, \text{home}) = \frac{P(\text{skips}, \text{unknown}, \text{followUp}, \text{short}, \text{home})}{P(\text{skips}, \text{unknown}, \text{followUp}, \text{short}, \text{home}) + P(\text{reads}, \text{unknown}, \text{followUp}, \text{short}, \text{home})}$$

For the numerator and denominator we have

$$\begin{aligned} P(\text{read})P(\text{unknown}|\text{read})P(\text{followUp}|\text{read})P(\text{short}|\text{read})P(\text{home}|\text{read}) &= \frac{9}{18} \cdot \frac{1}{3} \cdot \frac{2}{9} \cdot \frac{9}{9} \cdot \frac{4}{9} \\ &= 0.0165; \\ P(\text{skips})P(\text{unknown}|\text{skips})P(\text{followUp}|\text{skips})P(\text{short}|\text{skips})P(\text{home}|\text{skips}) &= \frac{9}{18} \cdot \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{2}{9} \cdot \frac{4}{9} \\ &= 0.0110, \end{aligned}$$

which gives

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

Estimating probabilities: zero probabilities

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 $\rightsquigarrow$ unreliable parameter estimates and risk of zero probabilities.

Estimating probabilities: zero probabilities

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 $\rightsquigarrow$ unreliable parameter estimates and risk of zero probabilities.

Using pseudo counts

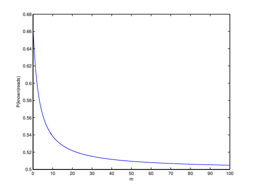
$$P(A = a|C = c) = \frac{N(A = a, C = c) + p_{ac} \cdot m}{N(C = c) + m}$$

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

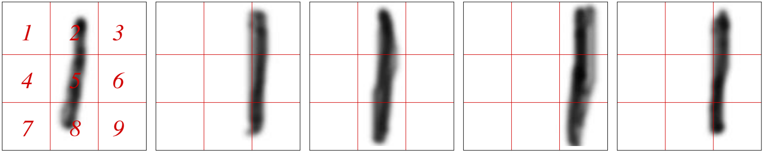
$$P(\text{known}|\text{reads}) = \frac{2 + 0.5 \cdot m}{3 + m}$$



The naive Bayes assumption I

Target: *Symbol* $\in \{A, \ldots, Z, 0, \ldots, 9\}$

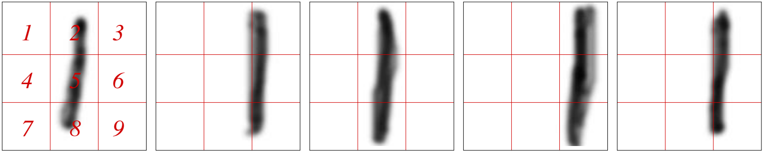
Predictors: *Cell-1*,...,*Cell-9* $\in \{b, w\}$.



The naive Bayes assumption I

Target: *Symbol* $\in \{A, \ldots, Z, 0, \ldots, 9\}$

Predictors: *Cell-1*,...,*Cell-9* $\in \{b, w\}$.



For example:

$$P(Cell-2=b \,|\, Cell-5=b, Symbol=1) > P(Cell-2=b \,|\, Symbol=1)$$

Attributes not independent given *Symbol*=1!

The naive Bayes assumption II

Target: *Spam* $\in \{y, n\}$

Predictors: *Subject-all-caps*, *Known-spam-server*, ..., *Contains'Money'* $\in \{y, n\}$.

The naive Bayes assumption II

Target: *Spam* $\in \{y, n\}$

Predictors: *Subject-all-caps*, *Known-spam-server*, ..., *Contains'Money'* $\in \{y, n\}$.

For example:

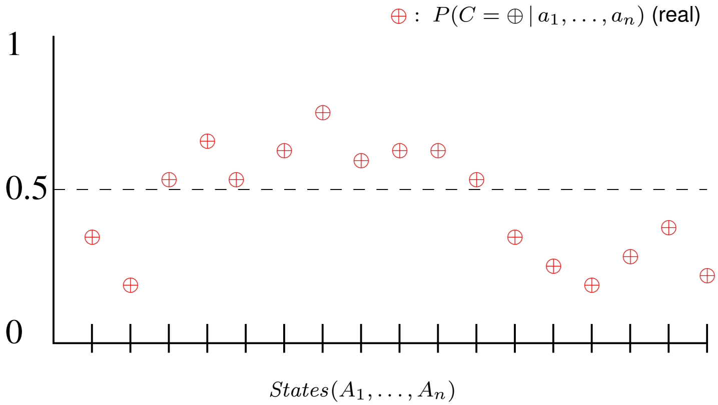
$$P(\text{Body'nigeria'=y} \mid \text{Body'confidential'=y}, \text{Spam=y}) \gg P(\text{Body'nigeria'=y} \mid \text{Spam=y})$$

Attributes not independent given *Spam*=yes!

$\leadsto$ Naive Bayes assumption often not realistic. Nevertheless, Naive Bayes often successful.

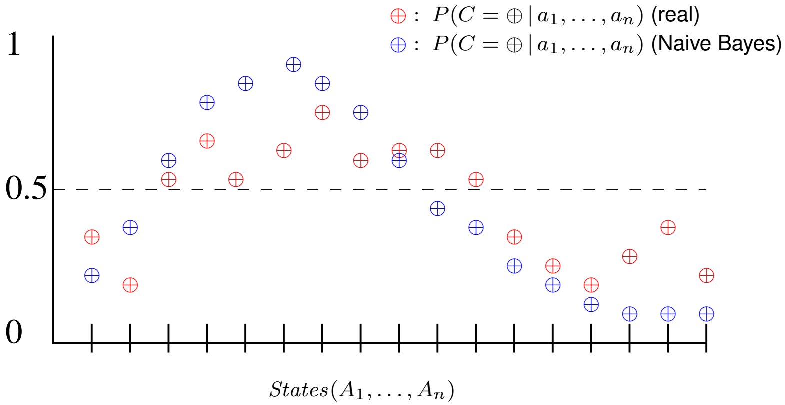
The paradoxical success of Naive Bayes

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]



When Naive Bayes must fail

No Naive Bayes Classifier can produce the following classification:

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

because assume it did, then:

$$\begin{array}{llcl}
1. & P(A=y \mid \oplus)P(B=y \mid \oplus)P(\oplus) & > & P(A=y \mid \ominus)P(B=y \mid \ominus)P(\ominus) \\
2. & P(A=y \mid \ominus)P(B=n \mid \ominus)P(\ominus) & > & P(A=y \mid \oplus)P(B=n \mid \oplus)P(\oplus) \\
3. & P(A=n \mid \ominus)P(B=y \mid \ominus)P(\ominus) & > & P(A=n \mid \oplus)P(B=y \mid \oplus)P(\oplus) \\
4. & P(A=n \mid \oplus)P(B=n \mid \oplus)P(\oplus) & > & P(A=n \mid \ominus)P(B=n \mid \ominus)P(\ominus)
\end{array}$$

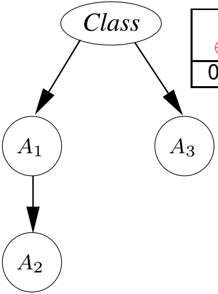
Multiplying the four left sides and the four right sides of these inequalities:

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

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

When features don't help

Data generated by process described by Bayesian network:



Class	A_1 = 0	A_1 = 1
$\oplus$	0.4	0.6
$\ominus$	0.5	0.5

A_1	A_2 = 0	A_2 = 1
0	1.0	0.0
1	0.0	1.0

Class $\oplus$	Class $\ominus$
0.5	0.5

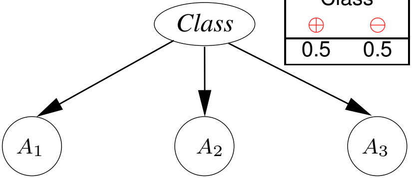
Class	A_3 = 0	A_3 = 1
$\oplus$	0.5	0.5
$\ominus$	0.7	0.3

Attribute A_2 is just a duplicate of A_1. Conditional class probability for example:

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

A word of caution

The Naive Bayes model learned from data:



Class	
$\oplus$	$\ominus$
0.5	0.5

Class	A_1 = 0	A_1 = 1
$\oplus$	0.4	0.6
$\ominus$	0.5	0.5

Class	A_2 = 0	A_2 = 1
$\oplus$	0.4	0.6
$\ominus$	0.5	0.5

Class	A_3 = 0	A_3 = 1
$\oplus$	0.5	0.5
$\ominus$	0.7	0.3

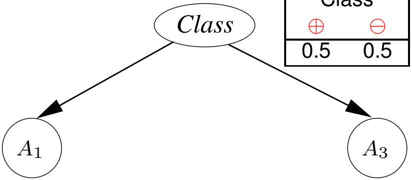
In Naive Bayes model:

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

Intuitively: the NB model double counts the information provided by A_1, A_2. Recall our previous discussion on the use of intermediate variables!

A word of caution

The Naive Bayes model with selected features A_1 and A_3:



Class	
$\oplus$	$\ominus$
0.5	0.5

	A_1	
Class	0	1
$\oplus$	0.4	0.6
$\ominus$	0.5	0.5

	A_3	
Class	0	1
$\oplus$	0.5	0.5
$\ominus$	0.7	0.3

In this Naive Bayes model:

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

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

Case-based Reasoning

Idea

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.

Distances for numeric features

If all features $\mathbf{X}$ are numeric:

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

Distances for discrete features

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

- 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:

	low	*medium*	*high*
low	0	2	5
medium	2	0	1
high	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 $d(\mathbf{x}, \mathbf{x}') = \sum_i w_i d_i(x_i, x'_i)$

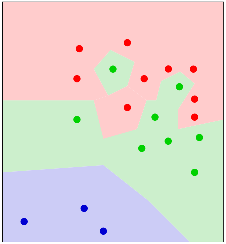
Given

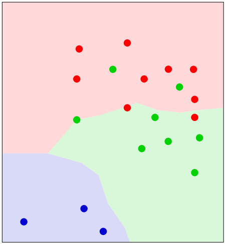
- training examples $(\mathbf{x}_i, y_i)$ $(i = 1, \ldots, N)$
- a new case $\mathbf{x}$ to be classified
- a distance measure $d(\mathbf{x}, \mathbf{x}')$

classify $\mathbf{x}$ as follows:

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

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

Overfitting

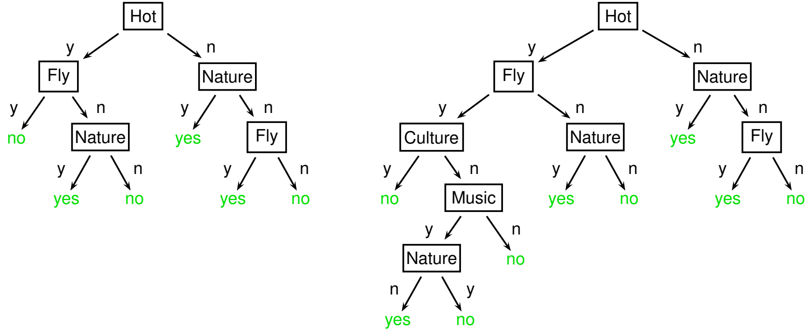
Holiday Data

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

Overfitting: Decision Trees

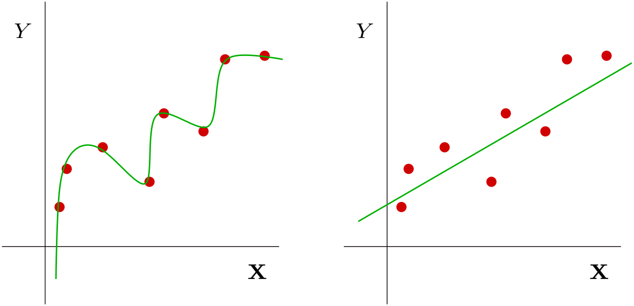
Decision tree learned from holiday data (left) and holiday data augmented with one more example

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?

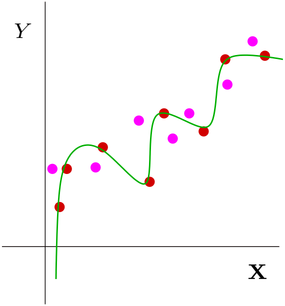
Overfitting: Neural Networks



Left model: minimizes SSE on training examples

Right model: may have smaller SSE on future observations

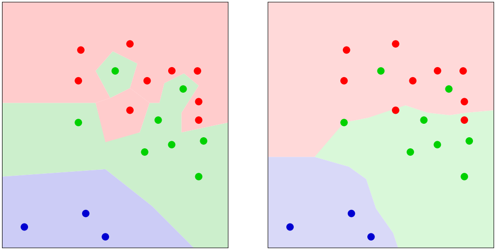
Overfitting: Neural Networks



Left model: minimizes SSE on training examples

Right model: may have smaller SSE on future observations

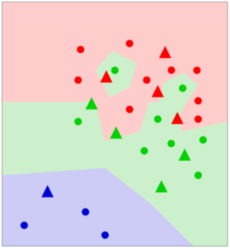
Overfitting: Nearest Neighbor



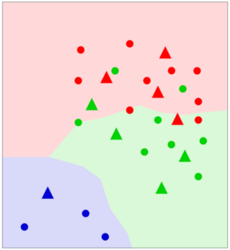
1-Nearest Neighbor 5-Nearest Neighbor

1-Nearest Neighbor correctly classifies all training examples

Overfitting: Nearest Neighbor



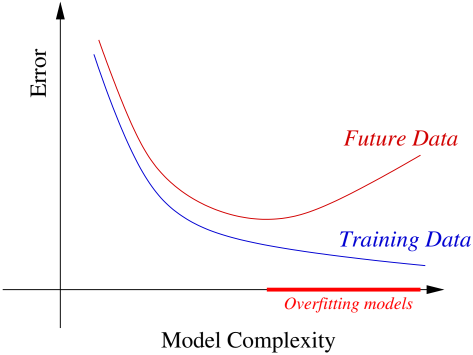
1-Nearest Neighbor



5-Nearest Neighbor

1-Nearest Neighbor correctly classifies all training examples

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



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.

Avoiding overfitting

Reduced hypothesis space

Do not allow overly complex models:

- Naive Bayes model
- K-NN for “large” K.

Modified Search/Scoring

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**

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

Validation data

Basic idea

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

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

Train/Validation split

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

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)

Selection of K

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

- measure accuracy of K-NN based on *training* examples for *validation* examples
- use K with best performance on *validation* examples
- *validation* examples can now be merged with *training* examples for predicting future cases

Example: K-NN and Neural Networks

Selection of K

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

- measure accuracy of K-NN based on *training* examples for *validation* examples
- 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,. . . , max, and #*nd* = 1,2,. . . , 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
- 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

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 $i = 1, \ldots, n$:
  - learn model (with given “complexity setting”) using folds $1, \ldots, i-1, i+1, \ldots, n$
  - 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