A Gentle Introduction to Machine Learning in Natural Language Processing using R

ESSLLI '2013
Düsseldorf, Germany

http://ufal.mff.cuni.cz/mlnlpr13

Barbora Hladká
hladka@ufal.mff.cuni.cz

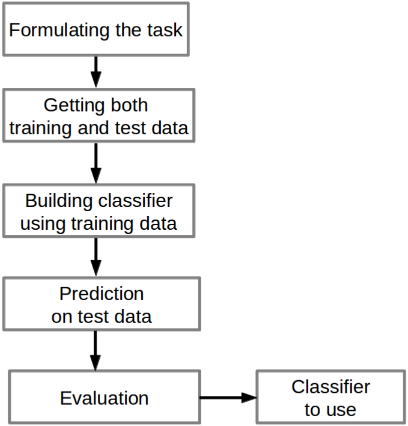
Martin Holub
holub@ufal.mff.cuni.cz

Charles University in Prague,
Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics

Day 3

- 3.1 Formal foundations of ML
- 3.2 Naive Bayes learning – Theory
- 3.3 Naive Bayes learning – Practice
- 3.4 Evaluation of a classifier
- Summary

Machine learning process – five basic steps

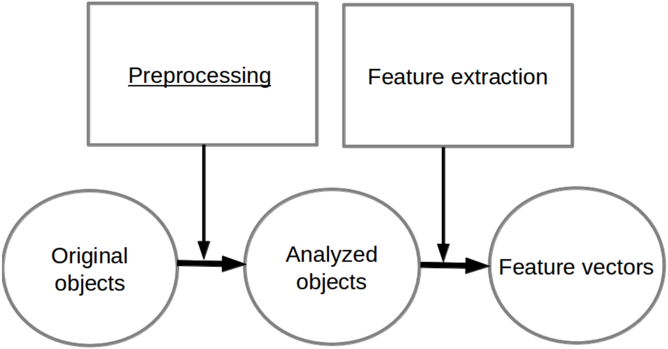


Formulating the task

1. **Task description**
   WSD: Assign the correct sense to the target word "line"
   COL: Decide whether the given word pair forms a semantic collocation
2. **Object specification**
   WSD: Sentences containing the target word
   COL: Word pairs
3. **Specification of target class C and its values $y_1, y_2, ..., y_k$**
   WSD: `SENSE = {CORD,DIVISION,FORMATION,PHONE,PRODUCT,TEXT}`
   COL: `Class = {YES,NO}`

Getting both training and test data

Step 1: Getting feature vectors



Getting both training and test data

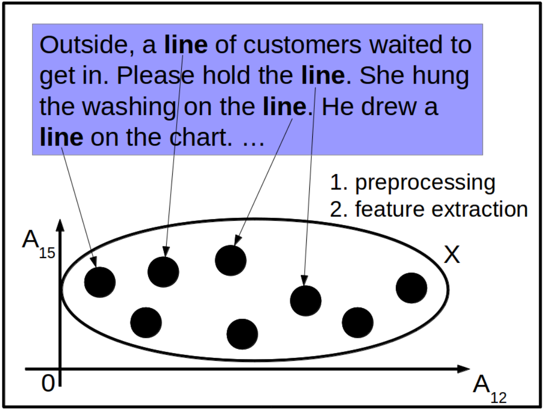
Step 1: Getting feature vectors

Notation

- Features as variables A_1, ..., A_m
- Feature values $x_1, ..., x_m$, $x_i \in A_i$
- Each object represented as feature vector $\mathbf{x} = \langle x_1, ..., x_m \rangle$
- feature vectors are elements in an *m*-dimensional feature space
- set of instances $X = \{\mathbf{x} : \mathbf{x} = \langle x_1, ..., x_m \rangle, x_i \in A_i\}$.

Getting both training and test data

Step 1: Getting feature vectors – Example

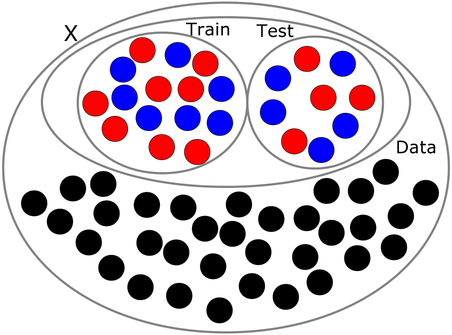


Getting both training and test data

Step 2: Assigning true classification

- Take *a number* of original objects and assign true classification to each of them.

- Take these objects and their true classification, do preprocessing and feature extraction. It results in $Data = \{\langle \mathbf{x}, y \rangle : \mathbf{x} \in X, y \in C\}$.



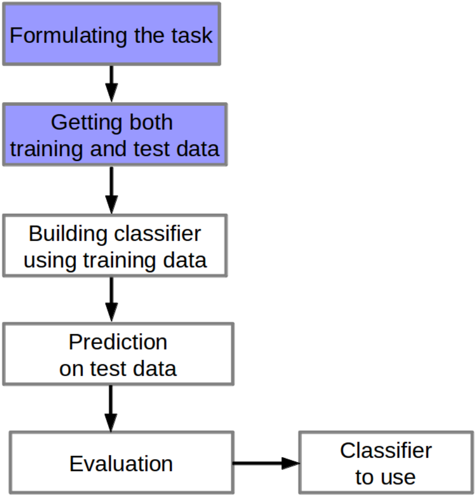
Getting both training and test data

Step 3: Selecting training set *Train* and test set *Test*

- $Train \subseteq Data$
- $Test \subseteq Data$
- $Train \cap Test = \emptyset$
- $Train \cup Test = Data$

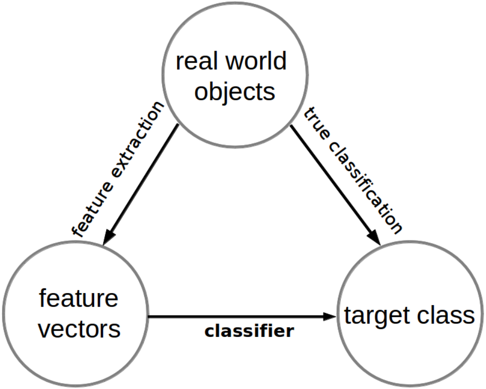
Machine learning process

Where are we now?



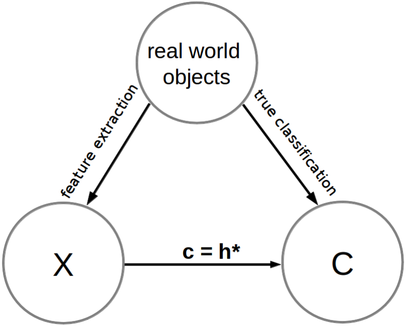
Building classifier

Classifier as a mapping



Building classifier

Classifier as a mapping



- We look for a **prediction function**, i.e. a classifier $c : X \rightarrow C : c(\mathbf{x}) = y$, $\mathbf{x} = \langle x_1, x_2, ..., x_m \rangle \in X, y \in C$.
- At the beginning we do not know the target prediction function. We need to approximate it using a **hypothesis** $h : X \rightarrow C$.
- Then, we **search** for the best hypothesis $h^\star$ that is finally taken as c.

Two types of parameters in machine learning

- Each machine learning method determines a particular form of **prediction function**.
- **The purpose of the learning process is to search for the best parameters of the prediction function**.

learning parameters	**hypothesis parameters**
= parameters of the learning process	= parameters of the prediction function

Building classifier using training data

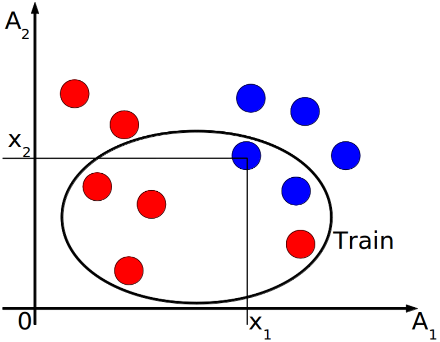
Terminological note

- **Model** = method + set of features + learning parameters
- **Classifier** = trained model, i.e. an output of the machine learning process, i.e. a particular method trained on a particular training data.
- **Prediction function** = classifier (used in mathematics). It's a function calculating a response value using predictor variables.
- **Hypothesis** = prediction function – not necessarily the best one (used in theory of machine learning).

Building classifier using training data

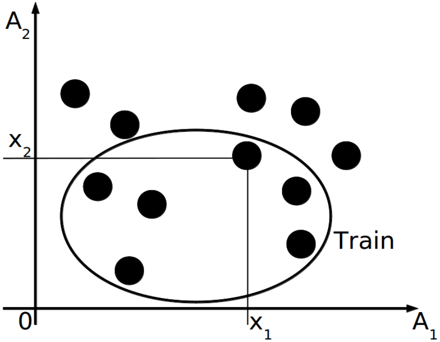
Supervised learning

$Data = \{\langle \mathbf{x}, y \rangle : \mathbf{x} \in X, y \in C\}$



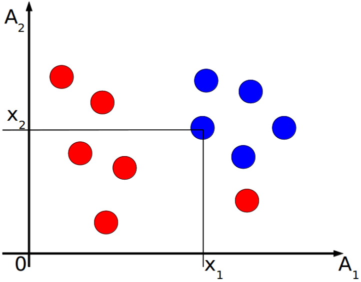
Unsupervised learning

$Data = \{\mathbf{x} : \mathbf{x} \in X\}$

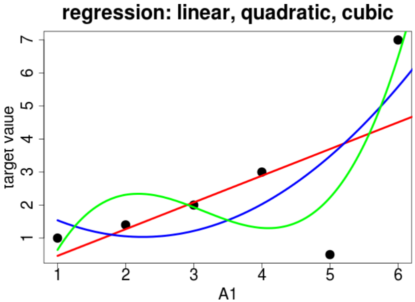


Building classifier using training data

Classification: C is categorical

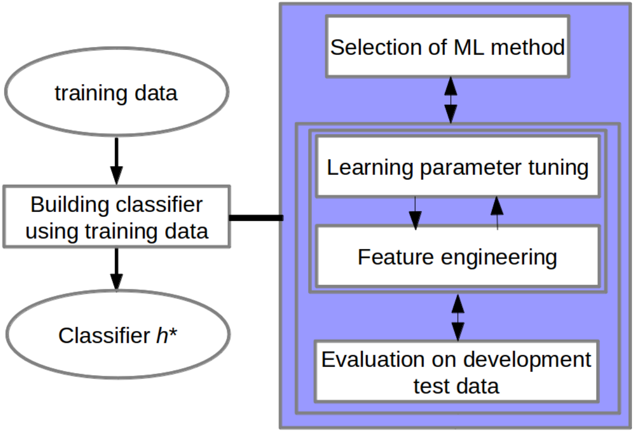


Regression: C is numerical



Building classifier using training data

Development cycle



Building classifier using development data

Development data is the set of all examples available to developer

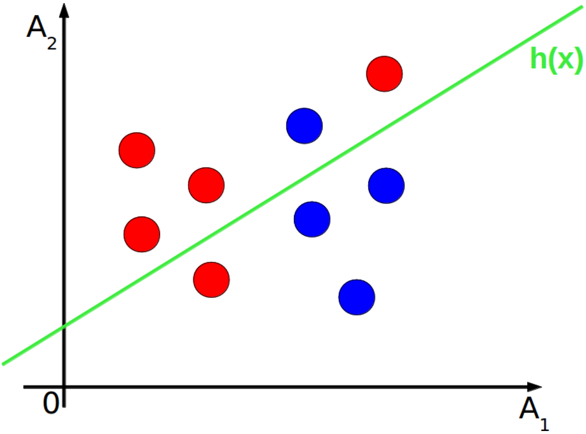
Development cycle

- **Input** Development data (e.g., `col.development.csv`)
  - Splitting the development data into development working set and development test set
- **Iteration**
  - Learning parameters setting and feature set selection
    Then using development working data to train a classifier
  - Prediction on development working and test sets
    Computing **training error** and **generalization error**
  - Evaluation and analysis of the current classifier
- **Output** $h^\star =$ the best classifier, with the lowest generalization error

Overfitting

Example 1

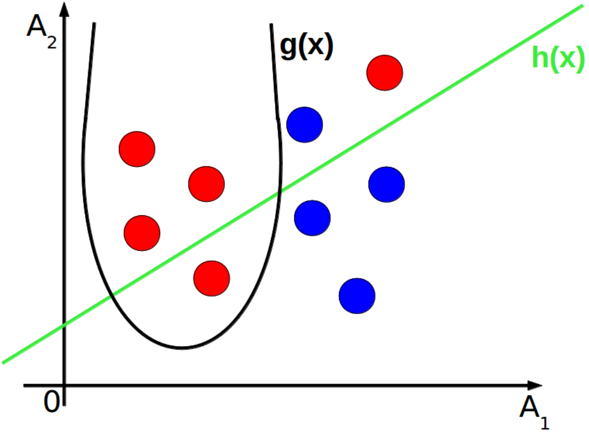
Draw decision boundary between classes described by a linear function $h(\mathbf{x})$



Overfitting

Example 2

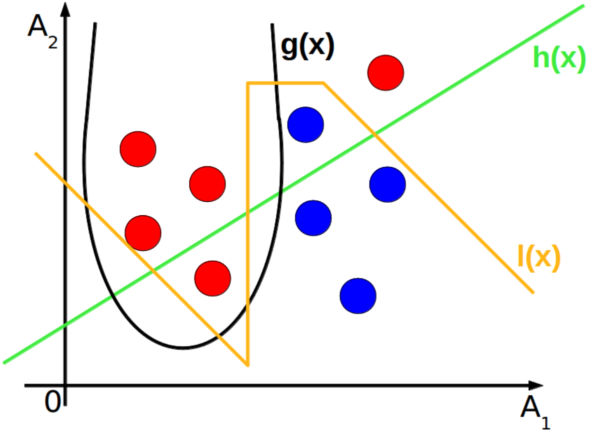
Draw decision boundary between classes described by quadratic function $h_2(\mathbf{x})$



Overfitting

Example 3

Draw decision boundary between classes described by complex function $h_3(\mathbf{x})$



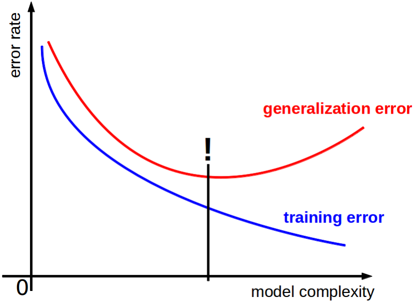
Overfitting

Comparing Examples 1–3

- $h(\mathbf{x})$: a straight line – determined by *two* parameters of the prediction function
  – doesn't fit two examples
- $h_2(\mathbf{x})$: a parabola – determined by *three* parameters of the prediction function
  – doesn't fit one example
- $h_3(\mathbf{x})$: a curve – determined by *many* parameters of the prediction function
  – perfectly fits all examples

Overfitting

If the generalization error increases while the training error steadily decreases then a situation of **overfitting** may have occurred.



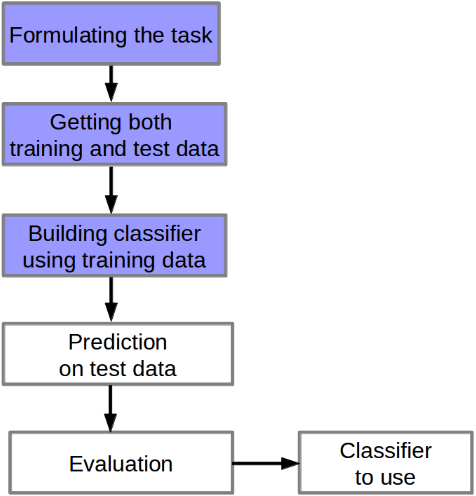
Generalization error has its global minimum $\Longrightarrow$ the best model

How to avoid overfitting

- feature engineering
  - **informative features**, i.e. useful for classification; control it by training error
  - **robust features**, i.e. not sensitive to training data; control it by generalization error
- learning parameters tuning

Machine learning process

Where are we now?

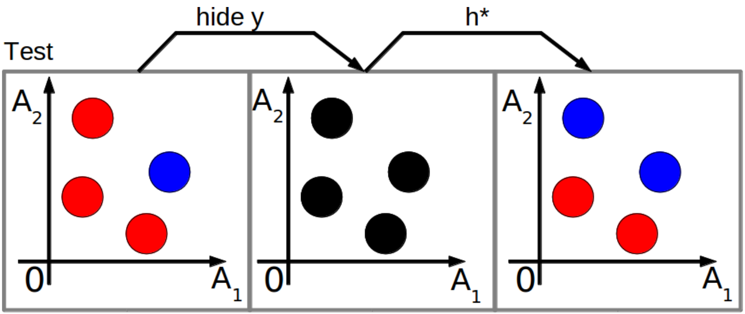


Prediction by $h^\star$ on test data

Test data *Test*, unseen during the training (e.g. `col.test.csv`)

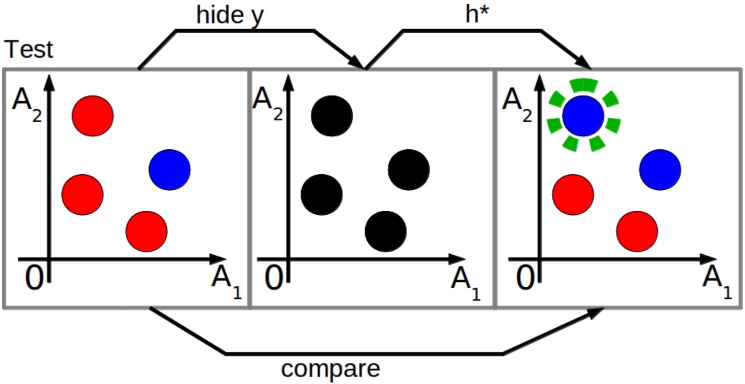
Doing prediction

$\forall \mathbf{x}$ such that $\langle \mathbf{x}, y \rangle \in Test$: Get $h^\star(\mathbf{x})$.

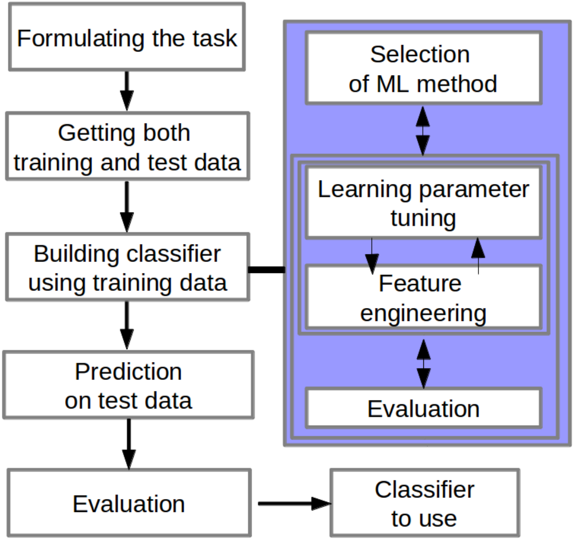


Evaluation of $h^\star$ on test data

Comparing true classification with the predicted classification
$\forall \mathbf{x}$ such that $\langle \mathbf{x}, y \rangle \in \mathit{Test}$: Compare y and $h^\star(\mathbf{x})$

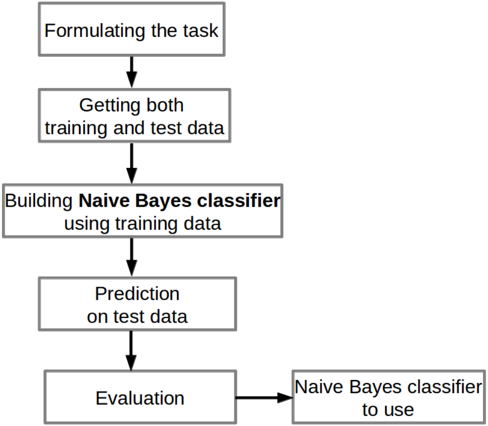


Machine learning process & Development cycle



Block 3.2
Naive Bayes learning – Theory

Machine learning process



Naive Bayes learning

Two types of parameters in machine learning – Examples

ML algorithm	**learning parameters**	**hypothesis parameters**
DT	`minsplit` (minimum number of instances in the associated training subset in order for a decision to be attempted), ...	decisions
NB	–	probabilities

Probability theory

Example The task of word sense disambiguation

- Outside, a **line** of customers waited to get in.
  - Are you sure of the sense `FORMATION`? Yes, I'm sure.
- He quoted a few **lines** from Shakespeare.
  - Are you sure of the sense `TEXT`? Yes, I'm sure.
- This has been a very popular new **line**.
  - Are you sure of the sense `PRODUCT`? No, I'm not sure.
  - Are you sure of the sense `CORD`? No, I'm not sure.
  - Which sense is more likely?

Probability theory provides a framework for the quantification and manipulation of uncertainty.

What is the sense of a word in a sentence?

Use conditional probabilities.

1. P($\texttt{CORD}$|This has been a very popular new line.)
2. P($\texttt{DIVISION}$|This has been a very popular new line.)
3. P($\texttt{FORMATION}$|This has been a very popular new line.)
4. P($\texttt{PHONE}$|This has been a very popular new line.)
5. P($\texttt{PRODUCT}$|This has been a very popular new line.)
6. P($\texttt{TEXT}$|This has been a very popular new line.)

Output the sense with **the highest conditional probability**.

Use training data to get conditional probabilities.

Probabilistic inference

Let $\mathbf{x}$ be an instance with feature values $x_1, x_2, ..., x_m$ and C is a target class with possible values $\{y_1, y_2, ..., y_k\}$.

Goal: Classify $\mathbf{x}$ into one of k classes $\{y_1, y_2, ..., y_k\}$.

Output: Target class value y^* with the highest (maximal) conditional probability $\mathrm{P}(y_i|\mathbf{x})$, i.e.

$$\boxed{y^* = argmax_{y_i \in C} \mathrm{P}(y_i|\mathbf{x})}$$

The `argmax` operator will give y_i for which $\mathrm{P}(y_i|\mathbf{x})$ is maximal.

Probabilistic inference

$\mathsf{P}(y_i|\mathbf{x})$ **and** $\mathsf{P}(\mathbf{x}|y_i)$

Example: Assume instance $\mathbf{x} = \langle x_{11}, x_{13}, x_{15} \rangle$.

$$\mathsf{P}(\texttt{PRODUCT}|\mathrm{TRUE}, \mathrm{draw}, \mathrm{between}) \overset{\textit{from definition}}{=} \frac{\mathsf{P}(\texttt{PRODUCT}, \mathrm{TRUE}, \mathrm{draw}, \mathrm{between})}{\mathsf{P}(\mathrm{TRUE}, \mathrm{draw}, \mathrm{between})}$$

$$\mathsf{P}(\mathrm{TRUE}, \mathrm{draw}, \mathrm{between}|\texttt{PRODUCT}) \overset{\textit{from definition}}{=} \frac{\mathsf{P}(\texttt{PRODUCT}, \mathrm{TRUE}, \mathrm{draw}, \mathrm{between})}{\mathsf{P}(\texttt{PRODUCT})}$$

Probabilistic inference

How to calculate $\mathrm{P}(y_i|\mathbf{x})$?

Use Bayes theorem

$$\boxed{\mathrm{P}(A|B) \overset{definition}{=} \frac{\mathrm{P}(B|A) * \mathrm{P}(A)}{\mathrm{P}(B)}}$$

Then

$$\boxed{y = argmax_{y_i \in C} \mathrm{P}(y_i|\mathbf{x}) \overset{Bayes\ theorem}{=} argmax_{y_i \in C} \frac{\mathrm{P}(\mathbf{x}|y_i)\mathrm{P}(y_i)}{\mathrm{P}(\mathbf{x})}}$$

Naive Bayes learning

$$y = argmax_{y_i \in C} \frac{\mathrm{P}(\mathbf{x}|y_i)\mathrm{P}(y_i)}{\mathrm{P}(\mathbf{x})} \stackrel{\mathbf{x}=\langle x_1,\dots,x_m \rangle}{=} argmax_{y_i \in C} \frac{\mathrm{P}(x_1,\dots,x_m|y_i)\mathrm{P}(y_i)}{\mathrm{P}(x_1,\dots,x_m)}$$

- Since $\mathrm{P}(x_1,\dots,x_m)$ is not dependent on C, it doesn't influence $argmax_{y_i \in C}$. Therefore

$$\boxed{y = argmax_{y_i \in C} \mathrm{P}(x_1,\dots,x_m|y_i)\mathrm{P}(y_i)}$$

Naive Bayes learning

assumes that **features** it uses are **conditionally independent** of one another given a target class.

Formal definition of conditional independence

Two events A and B are conditionally independent given an event D if

$$\boxed{\mathrm{P}(A|B, D) = \mathrm{P}(A|D)}$$

.

I.e. knowledge of B's value doesn't affect our belief in the value of A, given a value of D.

Naive Bayes learning

How to calculate $\mathrm{P}(\mathbf{x}|y_i)$ given the assumption of conditional independence of features given a target class C?

$$\mathrm{P}(\mathbf{x}|y_i) = \mathrm{P}(x_1, x_2, ..., x_m|y_i)$$
$$\stackrel{chain\ rule}{=} \mathrm{P}(x_1|x_2, ..., x_m, y_i)\mathrm{P}(x_2|x_3, ..., x_m, y_i)...\mathrm{P}(x_m|y_i)$$
$$\stackrel{ass.\ conditional\ indp.}{=} \Pi_{j=1}^{m}\mathrm{P}(x_j|y_i)$$

Then

$$\boxed{y = argmax_{y_i \in C}\Pi_{j=1}^{m}\mathrm{P}(x_j|y_i)\mathrm{P}(y_i)}$$

Naive Bayes classifier

How to calculate $\mathrm{P}(x_j|y_i)$ **and** $\mathrm{P}(y_i)$**?**

From training set *Train* that contains n training examples ($|Train| = n$):

- probabilities of classes

$$\boxed{\mathrm{P}(y_i) = |\{\mathbf{x} : \langle \mathbf{x}, y_i \rangle \in Train|/n}$$

- conditional probabilities

$$\boxed{\mathrm{P}(x_j|y_i) = \frac{|\{\hat{\mathbf{x}} : \langle\langle \hat{x_1}, \hat{x_2}, ..., x_j, ..., \hat{x_m}\rangle, y_i\rangle \in Train\}|}{|\{\mathbf{x} : \langle \mathbf{x}, y_i\rangle \in Train|}}$$

Naive Bayes classifier

Naive assumption of feature conditional independence given a target class is **rarely true** in real world applications.

Nevertheless, Naive Bayes classifier surprisingly often shows good performance in classification.

Block 3.3
Naive Bayes (NB) classifier – Practice in R

The very basics of using Naive Bayes classifier implementation in R

- **Task**
  Assign the correct sense to the target word "line" ("lines", "lined")
- **Objects**
  Sentences containing the target word ("line", "lines", "lined")
- **Target class**
  $\texttt{SENSE} = \{\texttt{CORD}, \texttt{DIVISION}, \texttt{FORMATION}, \texttt{PHONE}, \texttt{PRODUCT}, \texttt{TEXT}\}$
- **Features**
  Binary features $A_1, A_2, ..., A_{11}$

NB classifier in R – preparing data

```
examples <- read.table("../data/wsd.development.csv", header=T)
examples$A1  <- as.factor(examples$A1)
examples$A2  <- as.factor(examples$A2)
examples$A3  <- as.factor(examples$A3)
 . . .
examples$A11 <- as.factor(examples$A11)

num.examples  <- nrow(examples)
num.train     <- round(0.9 * num.examples)
num.test      <- num.examples - num.train

set.seed(123); s <- sample(num.examples)

indices.train  <- s[1:num.train]
train          <- examples[indices.train,]
indices.test   <- s[(num.train+1):num.examples]
test           <- examples[indices.test,]
```

NB classifier in R – the package

First of all, if not installed yet, install the package e1071

```
# to install the package
> install.packages("e1071")

# to check if the package is installed
> library()

# to load the package
> library(e1071)

# to get help info
> help(naiveBayes)
```

NB classifier in R – learning model M1

The first model M1 uses only one feature, namely A4

```
# to create a Naive Bayes model
> M1 <- naiveBayes(SENSE ~ A4, data=train)
>
```

Prediction on training data

```
> P1 <- predict(M1, train[5], type="class")
> print(table(P1))
P1
     cord   division formation      phone    product       text
        0          0         0        150       3022          0
>
```

NB classifier in R – analyzing the model M1

Comparing the predicted values with the true senses

```
> table(train$SENSE, P1)
           P1
            cord division formation phone product text
  cord         0        0         0     0     303    0
  division     0        0         0     0     294    0
  formation    0        0         0     0     268    0
  phone        0        0         0   142     205    0
  product      0        0         0     0    1646    0
  text         0        0         0     8     306    0
>
```

56.37 % of training examples are predicted correctly

```
> round(sum(diag(table(train$SENSE, P1)))/num.train * 100, 2)
[1] 56.37
>
```

NB classifier in R – testing the model M1

Predicted values vs. true senses on the test data

```
> P1.test <- predict(M1, test[5], type="class")
> table(test$SENSE, P1.test)
          P1.test
           cord division formation phone product text
  cord        0        0         0     0      33    0
  division    0        0         0     0      28    0
  formation   0        0         0     0      28    0
  phone       0        0         0    12      21    0
  product     0        0         0     0     192    0
  text        0        0         0     1      37    0
```

57.95 % of test examples are predicted correctly

```
> round(sum(diag(table(test$SENSE, P1.test)))/num.test * 100, 2)
[1] 57.95
```

Using more features

More models are described in the attached R-script

Homework 3.1

1. Download the `col.development.csv` data set
2. Load it both into a spreadsheet and into R and look at the data
   - There are 10 numerical features and 1 categorical feature – the description is given on your handout material
3. Split the data into 90 % – 10 % training and test portions
4. Build your own classifier – you can use both (choose at least one)
   - Decision Tree classifier
   - Naive Bayes classifier

   – You can use any subset of the 11 features
5. Prepare a feedback for us – if you want

Block 3.4
Evaluation of a classifier

You need thorough evaluation to

1. **get a reliable estimate of the classifier performance**
   – i.e. how it will perform on new – so far unseen – data instances (possibly in the future)

2. **compare your classifiers** that you have developed
   – to decide which one is "the best"

= Model assessment and selection

Model robustness

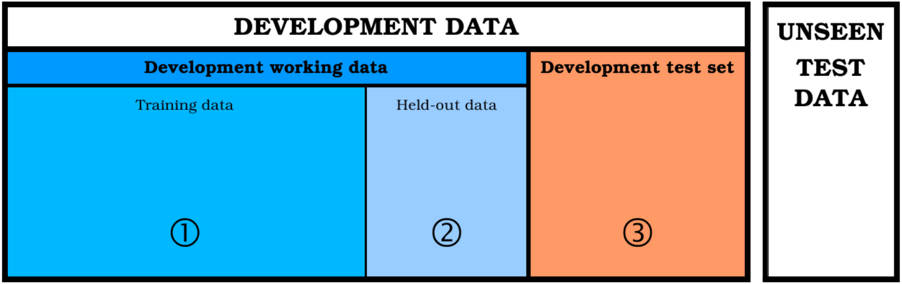
You need good performance

not only on *your* data,

but also on any data that can be expected!

Working with data

Working with data



Development data – the working portion

Development working data

Is used both for training your classifier and for evaluation when you tune the learning parameters.

- **Training data**
  is used for **training** your classifier with a particular learning parameter settings when you tune your classifier

- **Held-out data**
  is used for **evaluating** your classifier with a particular learning parameter settings when you tune your classifier

Development data – the test portion

Development test set

- the purpose is to simulate the “real” test data
- should be used only for your final development evaluation when your classifier has already been tuned and your learning parameters are finally set
- using it you get an estimate of your classifier’s performance at the end of the development
- is also used for model selection

Using bigger training sets

Generally, whenever you extend your training data, you should get a better classifier!

Using bigger training sets

Generally, whenever you extend your training data, you should get a better classifier!

If not, there is a problem

- either with your data
  – e.g. noise data or not representative data (distortion of statistical characteristics)
- or with your method/model
  – e.g. bad settings of learning parameters

– Sometimes, you cannot get better results because the performance is already stable/maximal. Even in this case using more training data should imply better robustness.

Using different training sets

1. When you tune your classifier you split your development working set and use only the "training portion" to train your classifier. You always hold out some data for classifier evaluation.
   In this phase you can do cross-validation, bootstrapping, or any other tricks. – Will be discussed later.
2. When you have your classifier tuned, keep the best parameters. Then use all "development working" portion as training data to make the best model.
3. Finally – after model selection – use all your development data as a training set to train the best model you are able to develop.
   This model can be later evaluated on the "unseen test" data (which is NOT a developer's job!).

Using a test set

- **Purpose** – How well will your classifier perform on novel data?
  – We can **estimate** the performance of the classifier using a test data set. And we do NOT have any better chance to get reliable estimate!
- Performance on the training data is not a good indicator of performance on future data.
  – You would easily **overestimate**!
- **Important!** – You should NOT have any **look** at your test data during the development phase!
  – Test set = independent instances that have NOT been used in **any way** to create the classifier.
- **Assumption** – Both training data and test data are representative samples of the underlying problem!

Baseline classifier

The most trivial baseline classifier is the classifier that always gives the most frequent class (sometimes called the MFC classifier).

Your classifier should never be worse than that baseline :–)

Usually a simple classifier (e.g. with a default settings of learning parameters) is considered to be a baseline. Then you compare your developed classifier to that baseline.

Confusion matrix

Confusion matrix is a square matrix indexed by all possible target class values.

```
** Comparing the predicted values with the true senses -- M3 **

           Prediction
Truth        cord division formation phone product text
  cord        268        3        10     7       9    6
  division      3      280         1     2       5    3
  formation    13        3       225     4      19    4
  phone        25        5         2   293      12   10
  product      51       10        39    32    1442   72
  text         12        1         7     4      28  262
```

Correctly predicted examples are displayed on the diagonal.

Accuracy and error rate

Accuracy
is the number of correctly predicted examples divided by the number of all examples in the predicted set

Error rate
is equal to **1 - accuracy**

The case of binary classification

Binary classification $\stackrel{aka}{=}$ 2-class classification $\stackrel{aka}{=}$ 0/1 classification

In binary classification tasks examples are sometimes regarded as divided into two disjoint subsets:

- **positive examples** – "to be retrieved" (ones)
- **negative examples** – "not to be retrieved" (zeros)

Confusion matrix for binary classification has only 4 cells

```
### Example confusion matrix for binary classification
> table(cv.test$Class, pred.test)
       prediction
          0   1
 true  0 580  69
       1  37 144
>
```

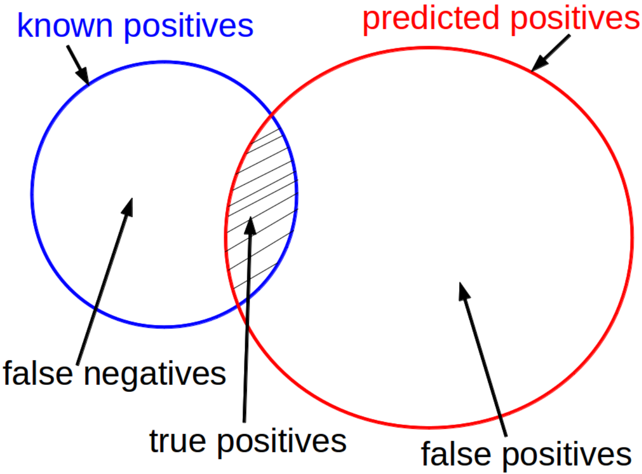
Confusion matrix for binary classification

		Predicted class	
		Positive	**Negative**
True class	**Positive**	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Explanation

- **'Trues'** are examples correctly classified
- **'Falses'** are examples incorrectly classified
- **'Positives'** were predicted as positives (correctly or incorrectly)
- **'Negatives'** were predicted as negatives (correctly or incorrectly)

Proportion of correctly predicted test examples



Basic performance measures

Measure	Formula
Precision	TP/(TP+FP)
Recall/Sensitivity	TP/(TP+FN)
Specifity	TN/(TN+FP)
Accuracy	(TP+TN)/(TP+FP+TN+FN)

Very often you need to **combine both good precision and good recall**. Then you usually use **balanced F-score**, so called **F-measure**

$$\boxed{F = 2\,\frac{\mathrm{Precision} * \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}}$$

Summary of Day 3