# INF5830 – 2013 FALL
## NATURAL LANGUAGE PROCESSING

Jan Tore Lønning, Lecture 15, 21.11

# Today

- Entropy
- Maximum entropy tagging
- Decision Trees
- A glimpse of non-linear classifiers and SVMs
- Combining classifiers
- Comparing classifiers

# Entropy



the average uncertainty of a single random variable

$$H(p) = H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$$

- $\log_2$ means  measuring in bits

**Uniform distribution**

| | a | b | c | d | entr. |
|---|---|---|---|---|---|
| p(x) | 1/4 | 1/4 | 1/4 | 1/4 | |
| log p(x) | -2 | -2 | -2 | -2 | |
| p(x)log p(x) | -1/2 | -1/2 | -1/2 | -1/2 | 2 |
| Optimal code | 11 | 10 | 00 | 01 | |

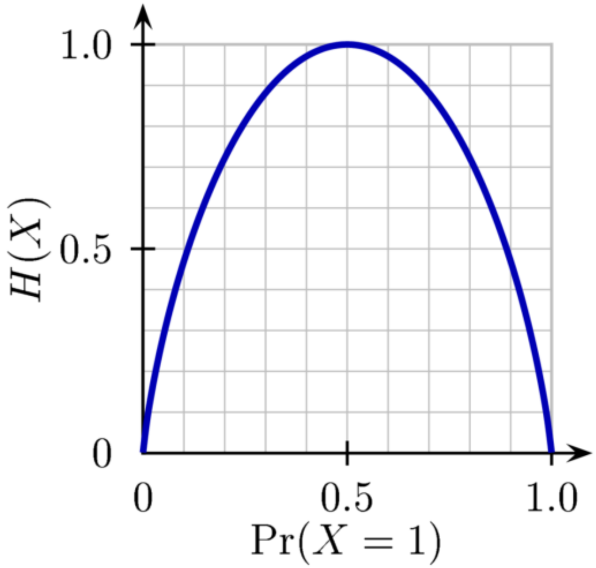
**Nonuniform distribution**

| a | b | c | d | entr. |
|---|---|---|---|---|
| 1/2 | 1/4 | 1/8 | 1/8 | |
| -1 | -2 | -3 | -3 | |
| -1/2 | -1/2 | -3/8 | -3/8 | 7/4 |
| 1 | 01 | 001 | 000 | |

# Binary entropy



- Tossing a fair coin:
  - Nothing is known of the outcome
  - Entropy = 1
- Throwing a dice, looking for 1/6:




$$H(p) = -\sum_{x \in X} p(x)\log_2 p(x) = -\frac{1}{6}\log_2\frac{1}{6} - \frac{5}{6}\log_2\frac{5}{6} \approx 0.65$$

# Today

- Entropy
- Maximum entropy tagging
- Decision Trees
- A glimpse of non-linear classifiers and SVMs
- Combining classifiers
- Comparing classifiers

# Multinomial logistic regression

- We may generalize this to more than two classes
  - For each class $c^j$ for j = 1,..,k
    - a linear expression $\vec{w}^j \bullet \vec{f} = \sum_{i=0}^{M} w_i^j x_i$
    - and the probability of belonging to class $c^j$:

$$P(c^j \mid \vec{f}) = \frac{1}{Z}\exp\left(\vec{w}^j \bullet \vec{f}\right) = \frac{1}{Z} e^{\vec{w}^j \bullet \vec{f}} = \frac{1}{Z} e^{\sum_i w_i^j f_i} = \frac{1}{Z}\prod_i \left(e^{w_i^j}\right)^{f_i} = \frac{1}{Z}\prod_i a_i^{f_i}$$

    - where $Z = \sum_{j=1}^{k} \exp\left(\vec{w}^j \bullet \vec{f}\right)$
    - and $a_i = e^{w_i^j}$

$$\frac{\text{Multinomial regression}}{\text{Logistic regression}} \approx \frac{\text{Naive Bayes (Bernoulli)}}{\text{Binary NB as linear classifier}}$$

# Indicator variables

$$P(c^j \mid \vec{f}) = \frac{1}{Z}\exp\left(\vec{w}^j \bullet \vec{f}\right) = \frac{\exp\left(\vec{w}^j \bullet \vec{f}\right)}{\sum_{l=1}^{k}\exp\left(\vec{w}^l \bullet \vec{f}\right)} = \frac{\exp\left(\sum_{i=0}^{n} w_i^j f_i\right)}{\sum_{l=1}^{k}\exp\left(\sum_{i=0}^{n} w_i^l f_i\right)} = \frac{\exp\left(\sum_{i=0}^{m} w_i\, f_i(c^j, x)\right)}{\sum_{l=1}^{k}\exp\left(\sum_{i=0}^{m} w_i\, f_i(c^l, x)\right)}$$

- Already seen: categorical variables represented by indicator variables, taking the values 0,1
- Also usual to let the variables indicate both observation and class

# Tagging

- Given a sequence of words $w_1^n = w_1 w_2 \cdots w_n$.
- Find the corresponding tag sequence $t_1^n$ which satisfies

$$\arg\max_{t_1^n} P\left(t_1^n \mid w_1^n\right)$$

# HMM tagging

$$\arg\max_{t_1^n} P(t_1^n \mid w_1^n) = \arg\max_{t_1^n} \frac{P(w_1^n \mid t_1^n)P(t_1^n)}{P(w_1^n)} = \arg\max_{t_1^n} P(w_1^n \mid t_1^n)P(t_1^n)$$

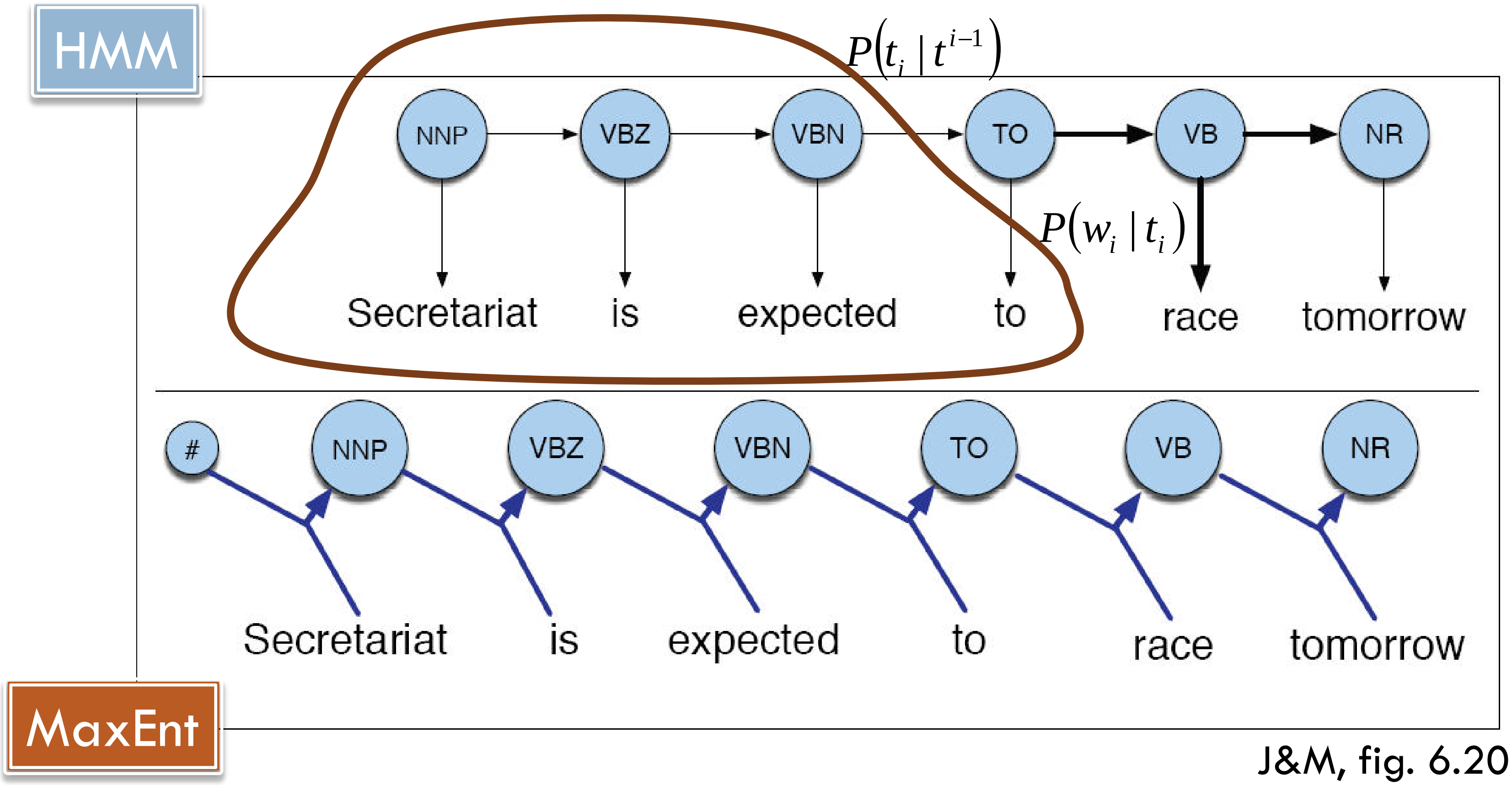
- HMM: simplifying assumptions:
  - Markov assumption for tags $P(t_1^n) = \prod_{i=1}^{n} P(t_i \mid t_0^{i-1}) \approx \prod_{i=1}^{n} P(t_i \mid t^{i-1})$
  - Local dependency between w and t:

$$P(w_1^n \mid t_1^n) = \prod_{i=1}^{n} P(w_i \mid w_1^{i-1}, t_1^n) \approx \prod_{i=1}^{n} P(w_i \mid t_i)$$

- Resulting expression

$$\arg\max_{t_1^n} P(t_1^n \mid w_1^n) = \arg\max_{t_1^n} \prod_{i=1}^{n} P(w_i \mid t_i) P(t_i \mid t_{i-1})$$

# Different strategies




J&M, fig. 6.20

Directly attacking: $\arg\max_{t_1^n} P(t_1^n \mid w_1^n)$

# MaxEnt tagging

$$P\left(t_1^n \mid w_1^n\right) = \prod_{i=1}^{n} P\left(t_i \mid t_1^{i-1}, w_1^n\right)$$

- At stage *i*
  - the history is an observation $h_i = t_1^{i-1}, w_1^n$
  - the tag $t_i$ is a class
- Example feature:
  - *$f_k(h_i, t_i)$=1 iff suffix ($w_i$)= "ing" and $t_i$=VBG, otherwise 0*
- *Ratnaparkhi restricts histories to*

$$h_i = \{w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}, t_{i-2}, t_{i-1}\}$$

- *Consider features from p.135*

# Maxent tagging decoding 1

- Ratnaparkhi: Beam search:
  - Tag from left to right
  - At stage *j* have a list of the *N* best hypotheses so far
    - Each hypothesis is a sequence of tags $t_1, t_2, \ldots, t_j$
  - At stage *j+1,*
    - for each (k = 1,…,N) hypothesis $(t\_k)_1^j$ consider all possible tags $t_{j+1}$ and calculate the probability of $(t_k)_1^j\ t_{j+1}$
    - keep the *N* best of these sequences

# Maxent tagging decoding 2

$$h_i = \{w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}, t_{i-2}, t_{i-1}\}$$

J&M: Maximum Entropy Markov Models

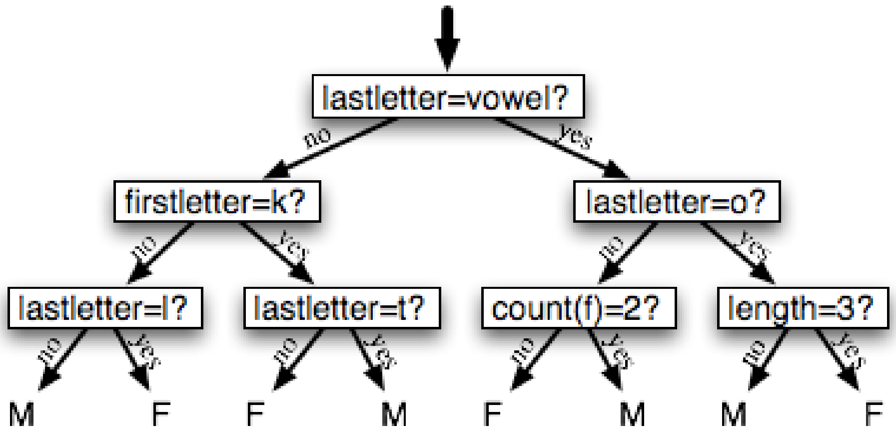
- Prerequisite: The tags included in the history must be restricted
  - Example: Ratnaparkhi's histories yield trigrams
- Use Viterbi for decoding:
  - After stage j:
    for each bigram of tags, a, b, there is one hypothesis $t_{ab}$ where $t_{j-1}$=a and $t_j$=b
  - At stage *j+1,*
    - For each pair of tags (b, c):
      - For all tags a: consider P(c|words, $t_{ab}$)P($t_{ab}$)
      - Choose the a=a' which yields the highest probability
      - Let $t_{bc}$ = $t_{a'b}$c and P($t_{bc}$)= P(c|words, $t_{ab}$)P($t_{ab}$)
  - Choose the best tag sequence at the end

# Today

- Entropy
- Maximum entropy tagging
- Decision Trees
- A glimpse of non-linear classifiers and SVMs
- Combining classifiers
- Comparing classifiers

# Decision trees






- Leave nodes are assigned classes
- Internal nodes correspond to features
- Daughters correspond to feature values
- Decoding: follow the tree

# Decision trees - construction



- In which order should the features be tested?
  1. Consider all "decision stumps"
     - (=a tree which only tests for one feature)
  2. Choose the optimal one, for some measure
  3. For nodes which have members from several classes, repeat the process
- Various measures for optimal stump, most common: Information gain:
  - which stump reduces entropy most?

# Example from WEKA

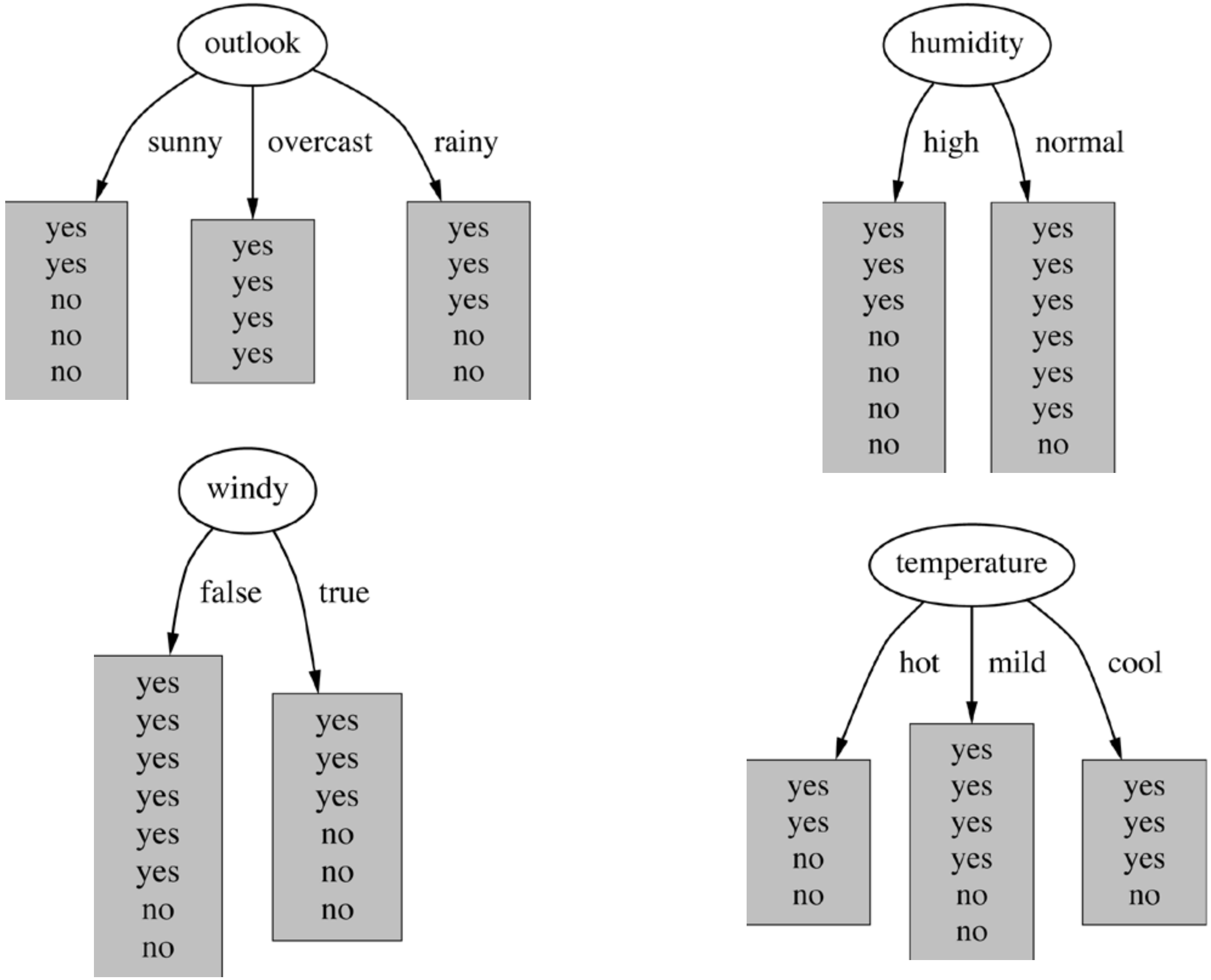


| Outlook | Temp | Humidity | Windy | Play |
|---|---|---|---|---|
| Sunny | Hot | High | False | No |
| Sunny | Hot | High | True | No |
| Overcast | Hot | High | False | Yes |
| Rainy | Mild | High | False | Yes |
| Rainy | Cool | Normal | False | Yes |
| Rainy | Cool | Normal | True | No |
| Overcast | Cool | Normal | True | Yes |
| Sunny | Mild | High | False | No |
| Sunny | Cool | Normal | False | Yes |
| Rainy | Mild | Normal | False | Yes |
| Sunny | Mild | Normal | True | Yes |
| Overcast | Mild | High | True | Yes |
| Overcast | Hot | Normal | False | Yes |
| Rainy | Mild | High | True | No |

$$H(p) = -\sum_{x \in X} p(x) \log_2 p(x) = -\frac{5}{14} \log_2 \frac{5}{14} - \frac{9}{14} \log_2 \frac{9}{14} \approx 0.94$$

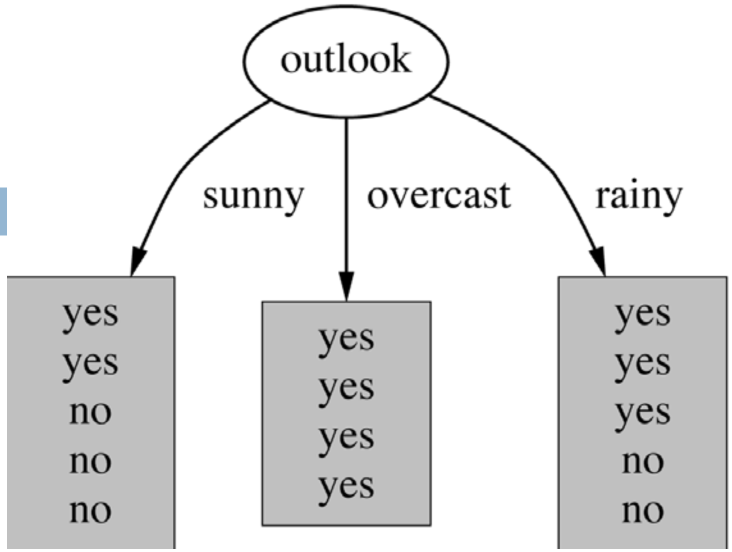
# Stumps




outlook
sunny
overcast
rainy
yes yes no no no
yes yes yes yes
yes yes yes no no
humidity
high
normal
yes yes yes no no no no
yes yes yes yes yes yes no
windy
false
true
yes yes yes yes yes yes no no
yes yes yes no no no
temperature
hot
mild
cool
yes yes no no
yes yes yes yes no no
yes yes yes no

# Info gain

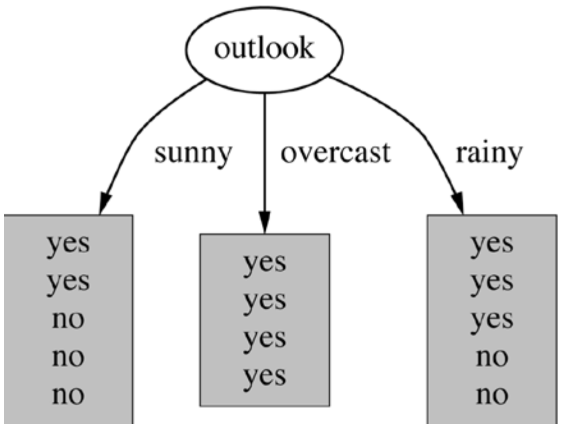


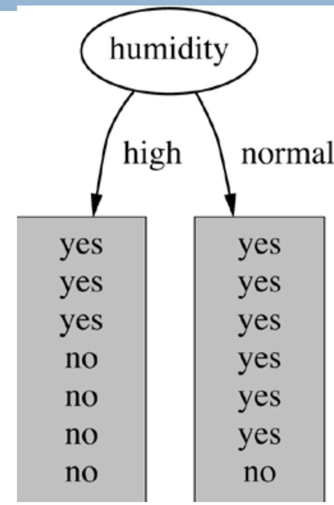


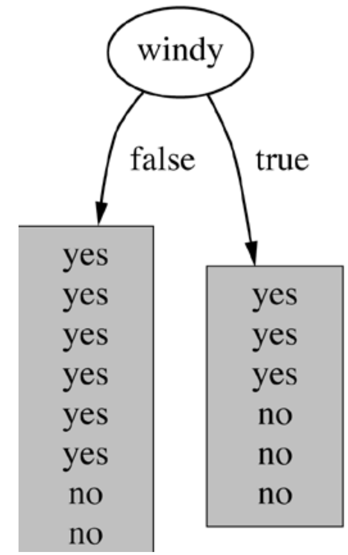

- Outlook = sunny $$H(p) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} \approx 0.97$$
- Outlook = overcast $$H(p) = -1\log_2 1 - 0\log_2 0 = 0$$
- Outlook = rainy $$H(p) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} \approx 0.97$$
- Average entropy $$\frac{5}{14}\times 0.97 + 0 + \frac{5}{14}\times 0.97 = 0.69$$
- gain(Outlook) $$0.94 - 0.69 = 0.25$$

# Stumps

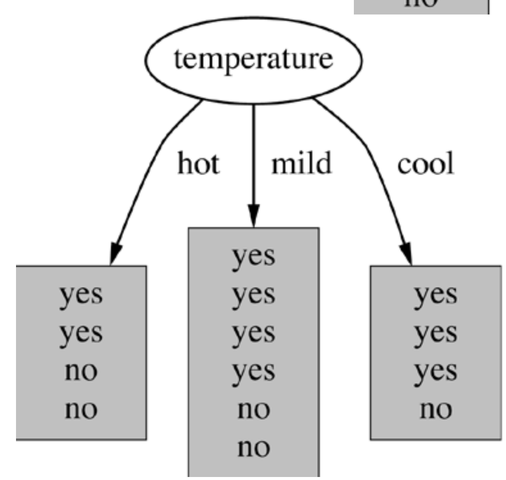




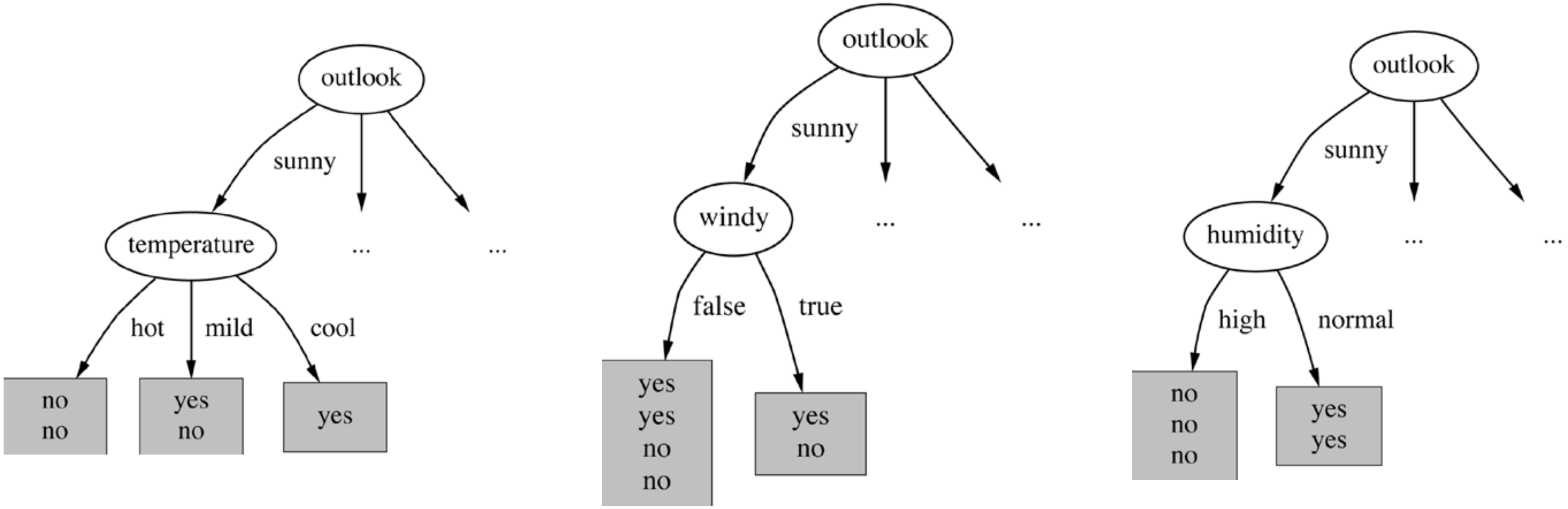




- gain(Outlook) 0.247
- gain(Humidity) 0.152
- gain(Windy) 0.048
- gain(Temp) 0.029



# Repeat

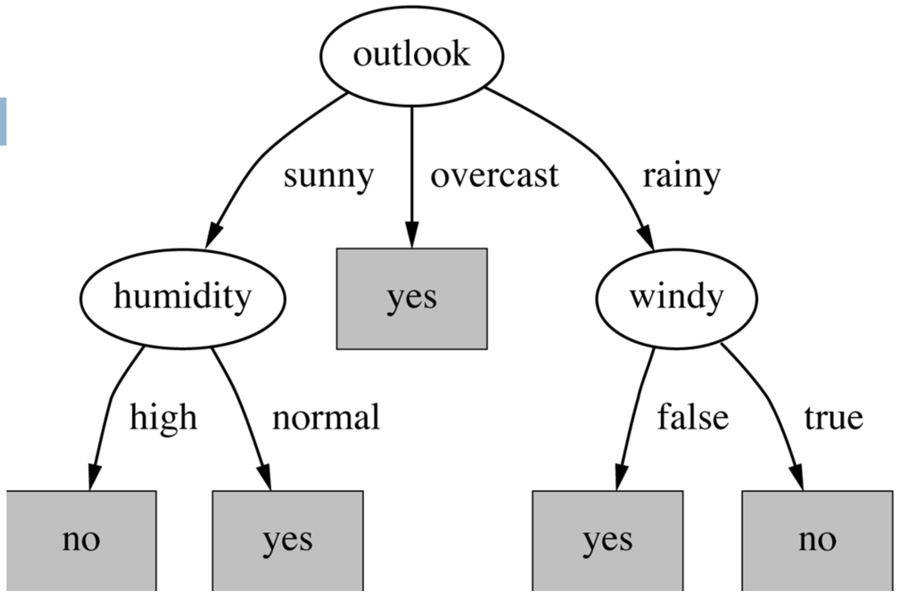





- gain(Temp) = 0.571
- gain(Humidity) = 0.971
- gain(Windy) = 0.020

# Final tree






- Stop when data can't be split further
- Leave nodes may be impure:
  - When decoding select the majority class of the  node
  - Or (for some pruposes) return the probability distribution of the node

# Danger for overfitting



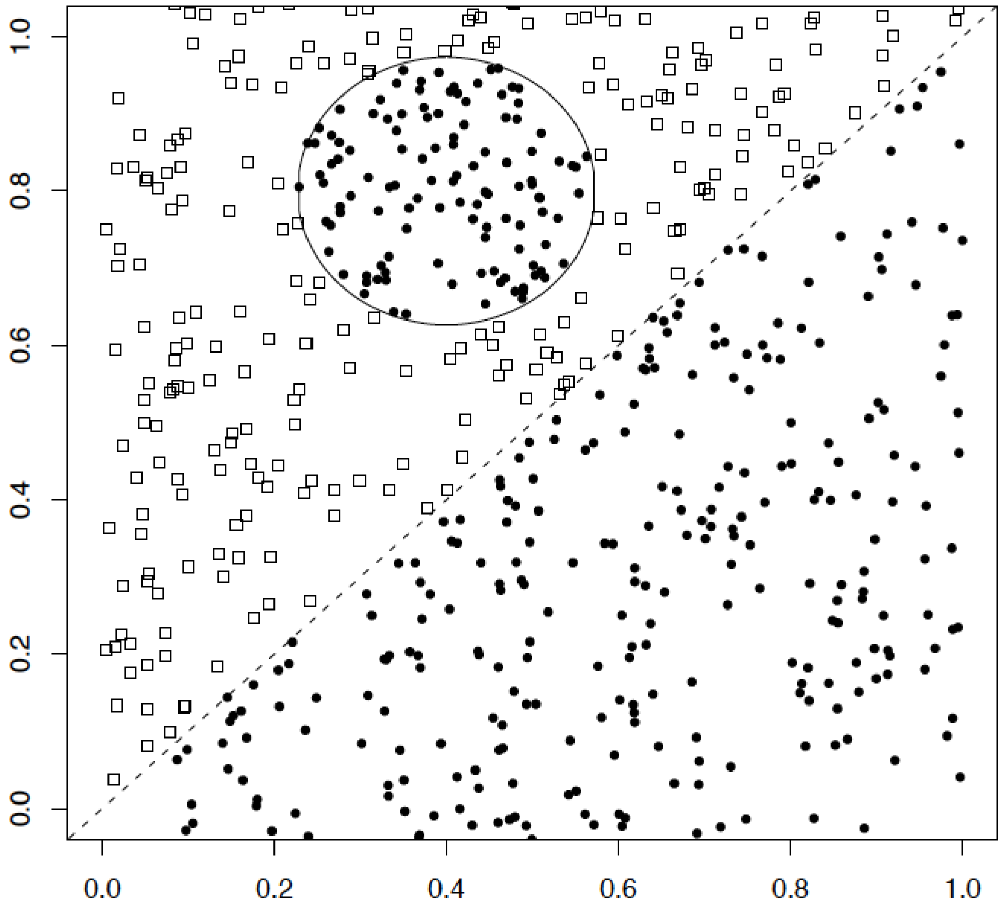
- Alt. 1: Stop splitting when the nodes correspond to little training data
- Alt. 2: Pruning:
  - Use development data
  - If there is no differrence (or little difference) between sister leaves, retract to mother

# Today

- Entropy
- Maximum entropy tagging
- Decision Trees
- A glimpse of non-linear classifiers and SVMs
- Combining classifiers
- Comparing classifiers

# A nonlinear problem

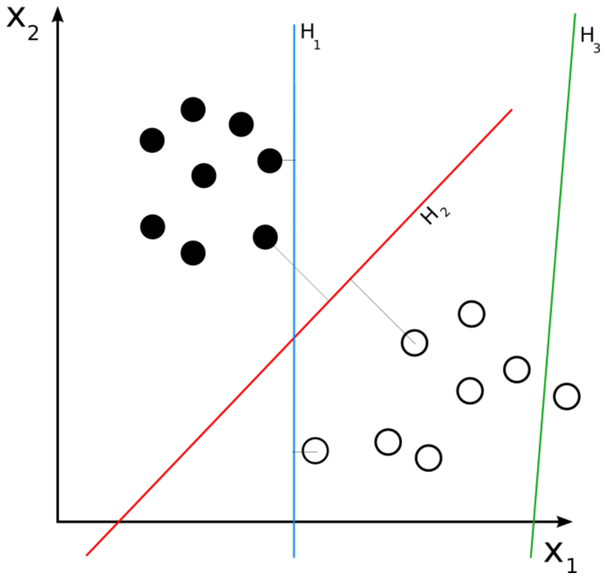





- A linear classifier like Naïve Bayes does badly on this task
- kNN will do very well (assuming enough training data)

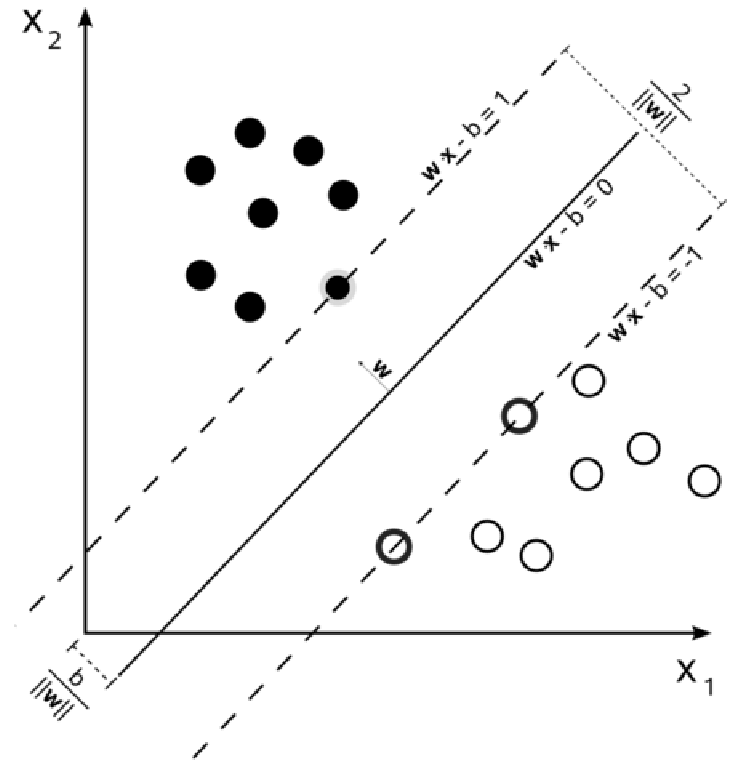
# Selecting hyperplanes

- If the training set is linearly separable, there are infinitely many separating hyperplanes.
- They all separate the training set
- But are not equally good on general test data
  - Perceptron – not so good
  - Naïve Bayes and Rocchio better
- Support Vector Machine (SVM) finds an optimal solution.
  - Maximizes the distance between the hyperplane and the “difficult points” close to decision boundary



# Support Vector Machine (SVM)

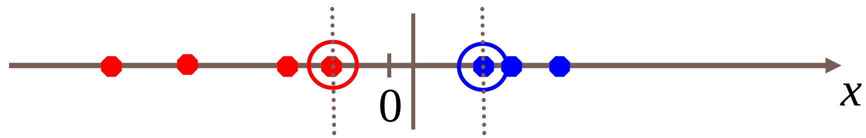
- SVMs maximize the *margin* around the separating hyperplane.
- The points in the training sets closest to the separating planes are called support vectors
- The decision function is specified by the support vectors.
- Currently widely seen as the best text classification method.



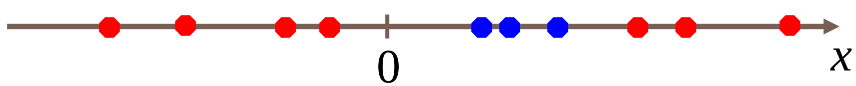
# Non-linear SVMs



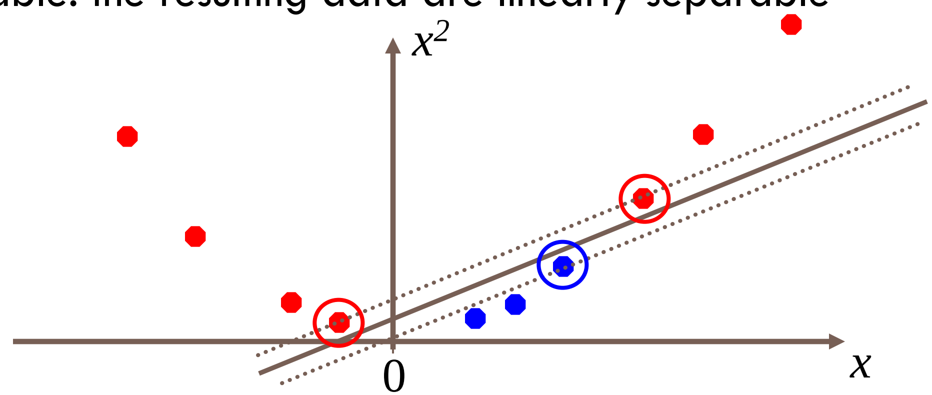
- Datasets that are linearly separable (with some noise) work out great:




- What to do if the datasets are not linearly separable?

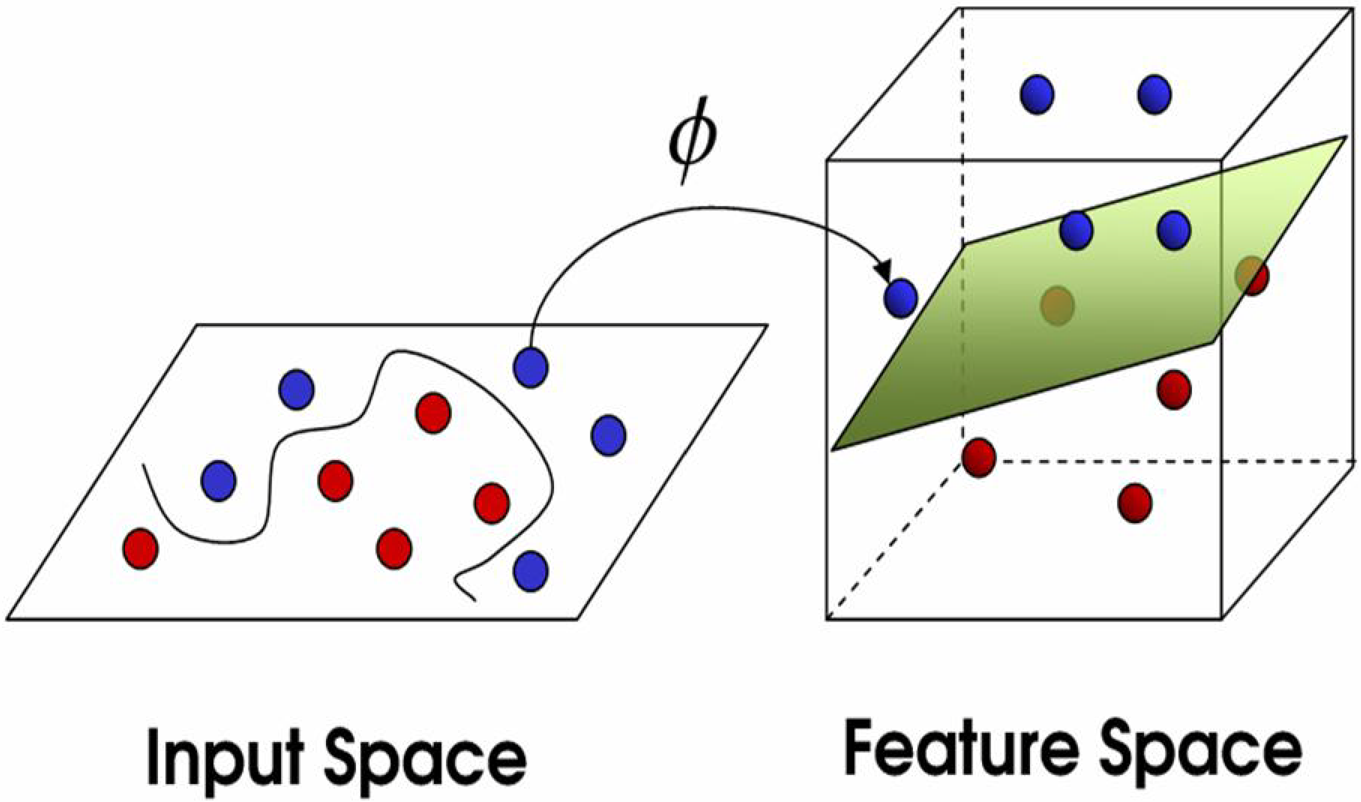



- Map data to a higher-dimensional space using some suitable mapping.
  - Suitable: the resulting data are linearly separable



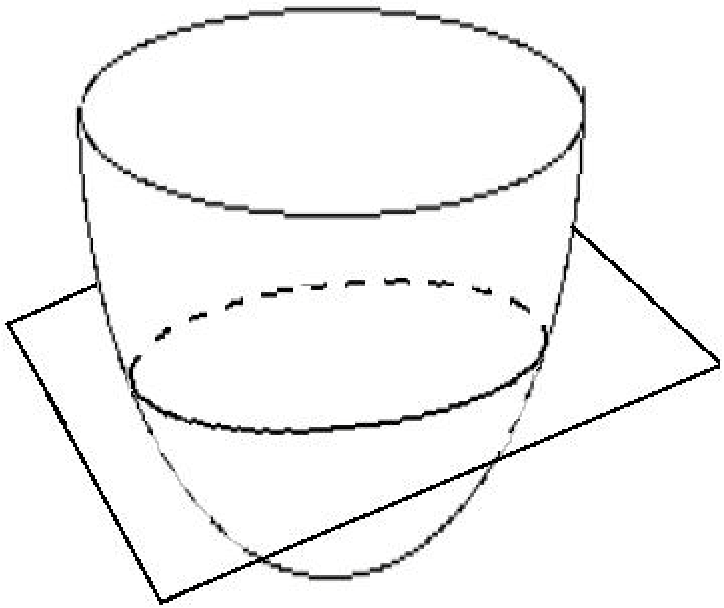

$x \rightarrow (x, x^2)$

# Principle of Support Vector Machines (SVM)



# SVMs – main ideas

- Maximize the distance between training data and a separating plane.
- Mapping a non-linear problem to a linear problem in higher dimensions using a kernel function.



# Today

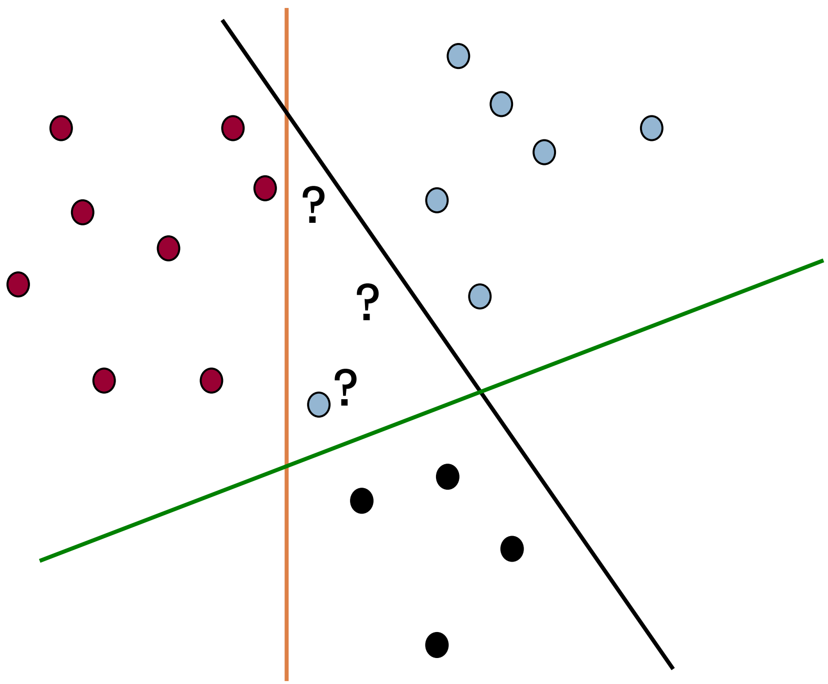
- Entropy
- Maximum entropy tagging
- Decision Trees
- A glimpse of non-linear classifiers and SVMs
- Combining classifiers
- Comparing classifiers

# More than two classes (in general)

- Any of or multivalue classification
  - An item may belong to 1, 0 or more than 1 classes
  - Classes are independent
  - Use *n* binary classifiers
  - Example: Documents
- One-of or multinomial classification
  - Each item belongs to one class
  - Classes are mutually exclusive
  - Example: POS-tagging

# One of classifiers

- Many classifiers are built for binary problems
- Simply combining several binary quantifiers do not result in a one-of-classifier.



# Combining binary classifiers

- Build a classifier for each class compared to its complement
- For a test document, evaluate it for membership in each class
- Assign document to class with either:
  - maximum probability
  - maximum score
  - maximum confidence
- Multinomial logistic regression is a good example
- Sometimes one postpones the decision and proceed with the probabilities (soft classification),
  - E.g. Maxent tagging

# Combining evaluation measures



**class 1**

| | truth: yes | truth: no |
|---|---|---|
| call: yes | 10 | 10 |
| call: no | 10 | 970 |

**class 2**

| | truth: yes | truth: no |
|---|---|---|
| call: yes | 90 | 10 |
| call: no | 10 | 890 |

**pooled table**

| | truth: yes | truth: no |
|---|---|---|
| call: yes | 100 | 20 |
| call: no | 20 | 1860 |

- Macroaverage:
  - Calculate accuracy/precision/recall for each class
  - Average over the classes (ignoring class size)
- Microaverage:
  - Pool the tables for all the classes
  - Calculate the accuracy/precision/recall for this class

# Today

- Entropy
- Maximum entropy tagging
- Decision Trees
- A glimpse of non-linear classifiers and SVMs
- Combining classifiers
- Comparing classifiers

# Maxent vs Naive Bayes

- If the Naive Bayes assumption is warranted – i.e. the features are independent – the two yield the same result in the limit.
- Otherwise, Maxent cope better with dependencies between features:
  - What happens in the two strategies if a feature gets repeated twice?

- With Maxent you may throw in features and let the model decide whether they are useful
- Maxent training is slower

# Repeating a feature

| Example | |
|---|---|
| P(c1)=0.5 | P(c2)=0.5 |
| P(a\|c1)=0.6 | P(a\|c2)=0.4 |
| P(b\|c1)=0.2 | P(b\|c2)=0.4 |

- Naive Bayes:
  - consider an observation containing a and c:
    - Which class is assigned if each feature is counted once?
    - Which class if *a* is counted twice and *b* once?

# Generative vs discriminative model

| Generative (e.g. NB) | Discriminative (e.g. Maxent) |
| --- | --- |
| $P(\underline{o},\underline{c})$ | … |
| $P(\underline{c}\mid\underline{o})$ | $P(\underline{c}\mid\underline{o})$ |
| $\mathrm{argmax}_c\ P(c\mid\underline{o})$ | $\mathrm{argmax}_c\ P(c\mid\underline{o})$ |
| $P(\underline{o})$ | |
| $\mathrm{argmax}_o\ P(o)$ | |
| $P(\underline{o}\mid\underline{c})$ | |

See NLTK book

# Which classifier do I use for a given classification problem?



- There is no learning method that is optimal for all classification problems.
  - because there is a tradeoff between bias and variance.
- Factors to take into account:
  - How much training data is available?
  - How simple/complex is the problem? (linear vs. nonlinear decision boundary)
  - How noisy is the data?
  - How stable is the problem over time?
    - For an unstable problem, it's better to use a simple and robust classifier.

# Learning algorithms

- In terms of actual computation, there are two types of learning algorithms.
  i. Simple learning algorithms that estimate the parameters of the classifier directly from the training data,
     - Examples: Naive Bayes, Rocchio, kNN
  ii. Iterative algorithms
     - Maxent
     - Support vector machines
     - Perceptron
- The best performing learning algorithms usually require iterative learning.

# Naive Bayes vs. other methods



(a)

| | NB | Rocchio | kNN | SVM |
|---|---|---|---|---|
| micro-avg-L (90 classes) | 80 | 85 | 86 | 89 |
| macro-avg (90 classes) | 47 | 59 | 60 | 60 |

(b)

| | NB | Rocchio | kNN | trees | SVM |
|---|---|---|---|---|---|
| earn | 96 | 93 | 97 | 98 | 98 |
| acq | 88 | 65 | 92 | 90 | 94 |
| money-fx | 57 | 47 | 78 | 66 | 75 |
| grain | 79 | 68 | 82 | 85 | 95 |
| crude | 80 | 70 | 86 | 85 | 89 |
| trade | 64 | 65 | 77 | 73 | 76 |
| interest | 65 | 63 | 74 | 67 | 78 |
| ship | 85 | 49 | 79 | 74 | 86 |
| wheat | 70 | 69 | 77 | 93 | 92 |
| corn | 65 | 48 | 78 | 92 | 90 |
| micro-avg (top 10) | 82 | 65 | 82 | 88 | 92 |
| micro-avg-D (118 classes) | 75 | 62 | n/a | n/a | 87 |

Evaluation measure: $F_1$

Naive Bayes does pretty well, but some methods beat it consistently (e.g., SVM).