CS140b: Feature Selection

Marie Meteer
Brandeis University
March 24, 2017

Slides from Fei Xia via James Pustejovsky.

The Cycle of Computational Linguistics

- We can study anything about language ...

1. Formalize some insights
2. Study the formalism mathematically
3. Develop & implement algorithms

Select the features!

4. Test on real data

+ Data Representation

- Data Types
  - Continuous
  - Categorical/Symbolic
    - Nominal – No natural ordering
    - Ordered/Ordinal
    - Special cases: Time/Date, Addresses, Names, IDs, etc.
- Normalization for continuous values (0-1 common)
  - What if data has skew, outliers, etc.
  - Standardization (z-score) – Transform the data by subtracting the average and then dividing by the standard deviation – allows more information on spread/outliers
  - Look at the data to make these and other decisions!

+ Feature Selection, Preparation, and Reduction

- Learning accuracy depends on the data!
  - *Is the data representative of future novel cases* - critical
  - Relevance
  - Amount
  - Quality
    - Noise
    - Missing Data
    - Skew
  - Proper Representation
  - How much of the data is labeled (output target) vs. unlabeled
  - Is the number of features/dimensions reasonable?
    - Reduction

+ Gathering Data

- Consider the task – What kinds of features could help
- Data availability
  - Significant diversity in cost of gathering different features
  - More the better (in terms of number of instances, not necessarily in terms of number of dimensions/features)
    - The more features you have the more data you need
  - Jitter – Increased data can help with overfit – handle with care!
- Labeled data is best
- If not labeled
  - Could set up studies/experts to obtain labeled data
  - Use unsupervised and semi-supervised techniques
    - Clustering
    - Active Learning, Bootstrapping, Oracle Learning, etc.

+ Feature Selection - Examples

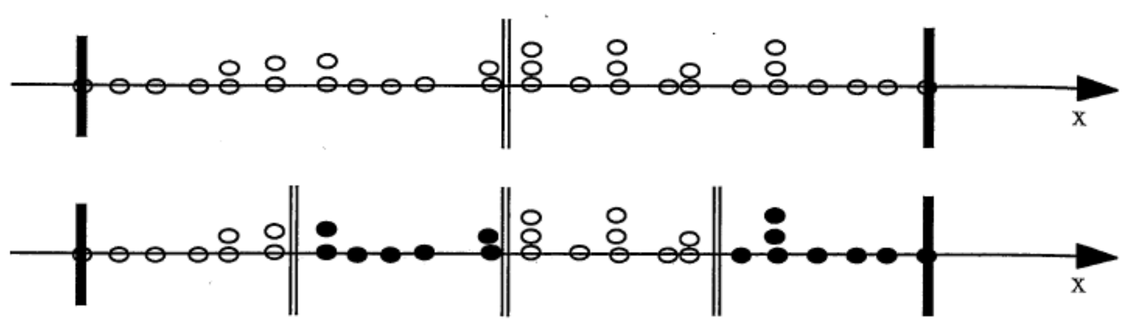
- Invariant Data
  - For character recognition: Size, Rotation, Translation Invariance
    - Especially important for visual tasks
  - Chess board features
    - Is vector of board state invariant?
- Character Recognition Class Assignment Example
  - Assume we want to draw a character with an electronic pen and have the system output which character it is
  - Assume an MLP approach with backpropagation learning
  - What features should we use and how would we train/test the system?

+ Transforming Continuous to Ordered Data

- Some models are better equipped to handle nominal/ ordered data
- Basic approach is to discretize/bin the continuous data
  - How many bins – what are tradeoffs? – seek balance
  - Equal-Width Binning
    - Bins of fixed ranges
    - Does not handle skew/outliers well
  - Equal-Height Binning
    - Bins with equal number of instances
    - Uniform distribution, can help for skew and outliers
    - More likely to have breaks in high data concentrations
  - Clustering
    - More accurate, though more complex
  - Bin borders are always an issue

+ Supervised Binning

- The previous binning approaches do not consider the classification of each instance and thus they are unsupervised (Class-aware vs. Class-blind)
- Could use a supervised approach which attempts to bin such that learning algorithms may more easily classify
- Supervised approaches can find bins while also maximizing correlation between output classes and values in each bin
  - Often rely on information theoretic techniques



+ Relevant Data

- Typically do not use features where
  - Almost all instance have the same value (no information)
    - If there is a significant, though small, percentage of other values, then might still be useful
  - Almost all instances have unique values (SSN, phone-numbers)
    - Might be able to use a variation of the feature (such as area code. month) or automatic transformation (e.g. season)
  - The feature is highly correlated with another feature
    - In this case the feature may be redundant and only one is needed
  - Careful if feature is too highly correlated with the target
    - Check this case as the feature may just be a synonym with the target and will thus lead to overfitting (e.g. the output target was bundled with another product so they always occur together)

+ Missing Data

- Need to consider approach for learning and execution (could differ)
- Throw out data with missing attributes
  - Could lose a significant amount of training set
  - Missing attribute may contain important information, (didn't vote can mean something about congressperson, extreme measurements aren't captured, etc.).
  - Doesn't work during execution
- Set (impute) attribute to its mode/mean based on rest of data set (too big an assumption?)
- Set attribute to its mode/mean given the output class (only works for training)
- Use a learning scheme (NN, DT, etc) to impute missing values
  - Train imputing models with a training set which has the missing attribute as the target and the rest of the attributes (including the original target) as input features. Better accuracy, though more time consuming - multiple missing values?
- Impute based on the most similar complete instance(s) in the data set
- Train multiple reduced input models to handle common cases of missing data
- Let unknown be just another attribute value – Can work well in many cases
  - Natural for nominal data
  - With continuous data, can use an indicator node, or a value which does not occur in the normal data (-1, outside range, etc.), however, in the latter case, the model will treat this as an extreme ordered feature value and may cause difficulties

+ Dirty Data and Data Cleaning

- Dealing with bad data, inconsistencies, and outliers
- Many ways errors are introduced
  - Measurement Noise/Outliers
  - Poor Data Entry
  - User lack of interest
    - Most common birthday when B-day mandatory: November 11, 1911
    - Data collectors don't want blanks in data warehousing so they may fill in (impute) arbitrary values
- Data Cleaning
  - Data analysis to discover inconsistencies
  - Noise/Outlier removal – Requires care to know when it is noise and how to deal with this during execution – Our experiments show outlier removal during training increases subsequent accuracy.
  - Clustering/Binning can sometimes help

+ Labeled and Unlabeled Data

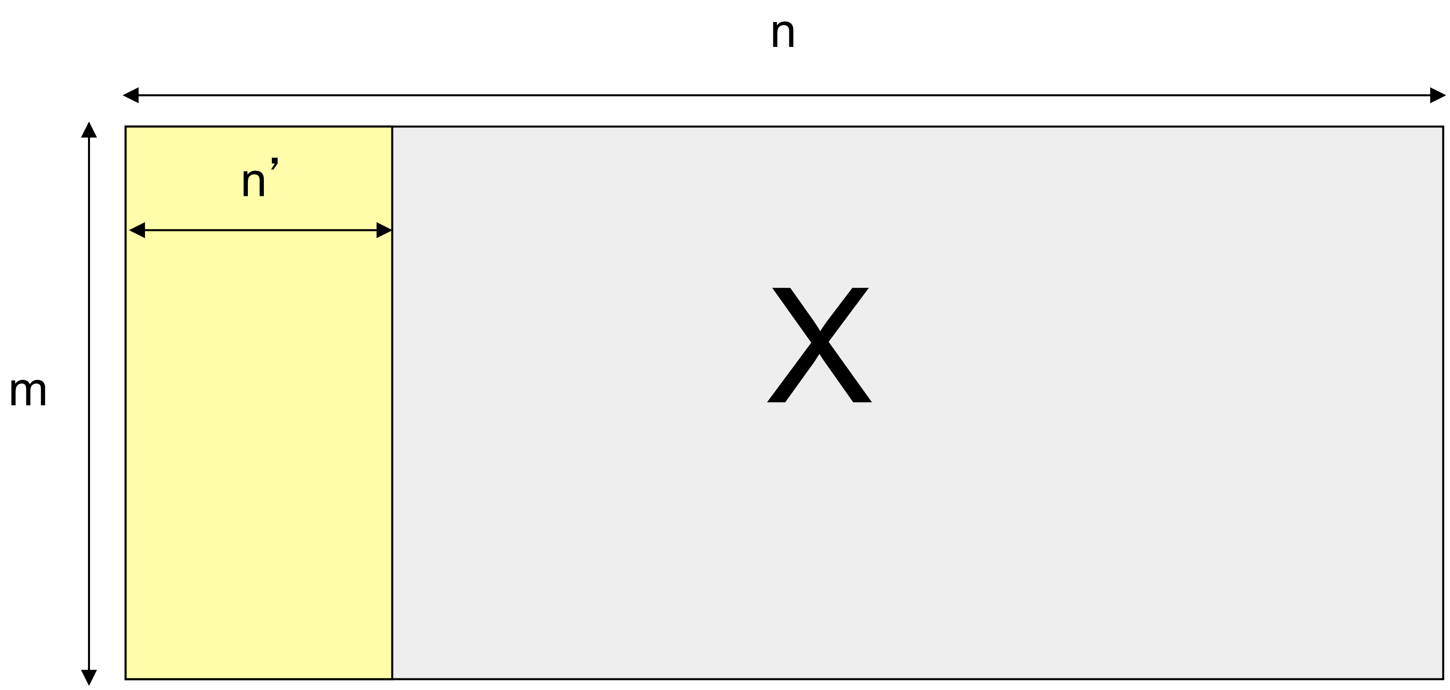
- Accurately labeled data is always best
- Often there is lots of cheaply available unlabeled data which is expensive/difficult to label – internet data, etc.
- Semi-Supervised Learning – Can sometimes augment a small set of labeled data with lots of unlabeled data to gain improvements
- Active Learning – Out of a large collection of unlabeled data, interactively select the next most informative instance to label
- Bootstrapping: Iteratively use current labeled data to train model, use the trained model to label the unlabeled data, then train again including most confident newly labeled data, and re-label, etc., until some convergence
- Combinations of above and other techniques being proposed

Feature Selection and Feature Reduction

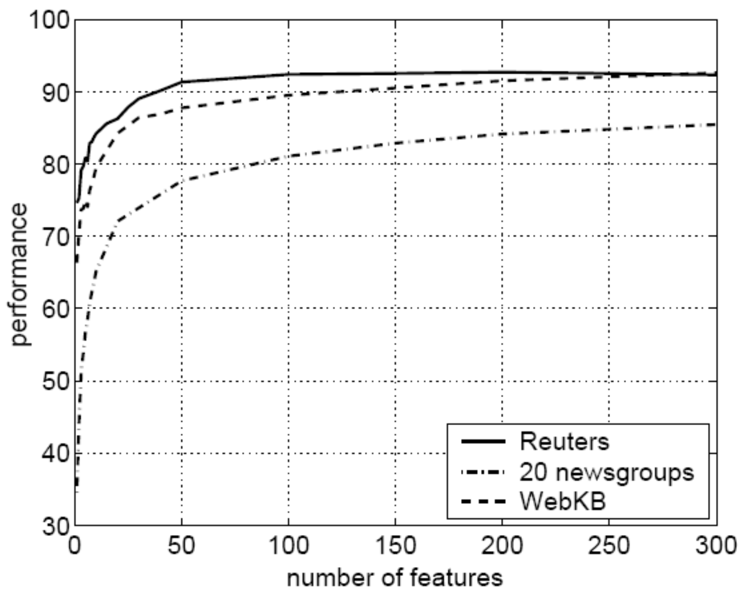
- Given n original features, it is often advantageous to reduce this to a smaller set of features for actual training
  - Can improve/maintain accuracy if we can preserve the most relevant information while discarding the most irrelevant information
  - and/or Can make the learning process more computationally and algorithmically manageable by working with less features
  - Curse of dimensionality requires an exponential increase in data set size in relation to the number of features to learn without overfit – thus decreasing features can be critical
- Feature Selection seeks a subset of the n original features which retains most of the relevant information
  - Filters, Wrappers
- Feature Reduction combines the original features into a new smaller set of features which hopefully retains most of the relevant information from all features - Data fusion (e.g. LDA, PCA, etc.)

+ Feature Selection

- **Thousands to millions of low level features**: select the most relevant one to build **better, faster, and easier to understand** learning machines.



+ Text Filtering



Reuters: 21578 news wire, 114 semantic categories.

20 newsgroups: 19997 articles, 20 categories.

WebKB: 8282 web pages, 7 categories.

Bag-of-words: >100000 features.

Top 3 words of some categories:

- **Alt.atheism**: atheism, atheists, morality
- **Comp.graphics**: image, jpeg, graphics
- **Sci.space**: space, nasa, orbit
- **Soc.religion.christian**: god, church, sin
- **Talk.politics.mideast**: israel, armenian, turkish
- **Talk.religion.misc**: jesus, god, jehovah

Bekkerman et al, JMLR, 2003

+ Feature types

- Target
  - What you are trying to learn?
  - Consider complexity
    - 43 parts of speech or 118?
  - What is the “unit”
    - Word for sense disambiguation
    - Document for topic
    - Utterance for speech acts
- “Features”
  - Selected knowledge that is used to train the model
  - Must be something I can measure/count!
  - Some are more obvious than others

Which features to use?

Most crucial decision you’ll make!

1. Topic
   - Words, phrases, ?
2. Author
   - Stylistic features
3. Sentiment
   - Adjectives, ?
4. Spam
   - Specialized vocabulary

+ How to choose features

- Consider cost
  - Words vs. POS vs parse tree
- Observable/countable
- Differentiating
  - Remove “non-informative” terms from documents
- Questions to consider
  - Stemmed or surface form?
  - Single words or phrases?
  - Multiwords (pre-identified phrases)
  - Words or word classes?
  - Remove stop words?

Word Sense Disambiguation

- Supervised machine learning approach:
  - A training corpus of words tagged in context with their sense
  - Corpus is used to train a classifier that can tag words in new text
- Summary of what we need:
  - the **tag set** (“sense inventory”)
  - the **training corpus**
  - A set of **features** extracted from the training corpus
  - A **classifier**

+ Feature vectors

- A simple representation for each observation (each instance of a target word)
  - Vectors of sets of feature/value pairs
    - I.e. files of comma-separated values
  - These vectors should represent the window of words around the target

+ Collocational

- Position-specific information about the words in the window
- guitar and bass player stand
  - [guitar, NN, and, CC, player, NN, stand, VB]
  - $\text{Word}_{n-2,}$ $\text{POS}_{n-2,}$ $\text{word}_{n-1,}$ $\text{POS}_{n-1,}$ Word_{n+1} POS_{n+1}…
  - In other words, a vector consisting of
  - [position n word, position n part-of-speech…]

+ Word Similarity: Context vector

- Consider a target word *w*
- Suppose we had one binary feature f_i for each of the *N* words in the lexicon v_i
- Which means "word v_i occurs in the neighborhood of *w*"
- w=(f1,f2,f3,…,fN)
- If w=tezguino, v1 = bottle, v2 = drunk, v3 = matrix:
- w = (1,1,0,…)

Co-occurrence vectors based on dependencies

- For the word "cell": vector of NxR features
  - R is the number of dependency relations
- What do I need for this?

	subj-of, absorb	subj-of, adapt	subj-of, behave	...	pobj-of, inside	pobj-of, into	...	nmod-of, abnormality	nmod-of, anemia	nmod-of, architecture	...	obj-of, attack	obj-of, call	obj-of, come from	obj-of, decorate	...	nmod, bacteria	nmod, body	nmod, bone marrow
cell	1	1	1		16	30		3	8	1		6	11	3	2		3	2	2

+ Semantic Role Labeling

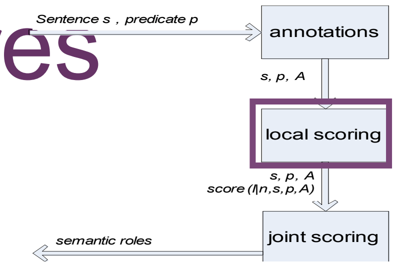
- What's the target? What am I trying to learn?
  - Traditional thematic roles
    - Agent, patient, theme, goal, instrument
  - FrameNet
    - Seller, buyer
  - "Agnostic" Propbank
    - A0, A1, A2
- What features are available that would help to model the distinctions?

Steps in SRL

From Xue & Palmer EMLNP 2004

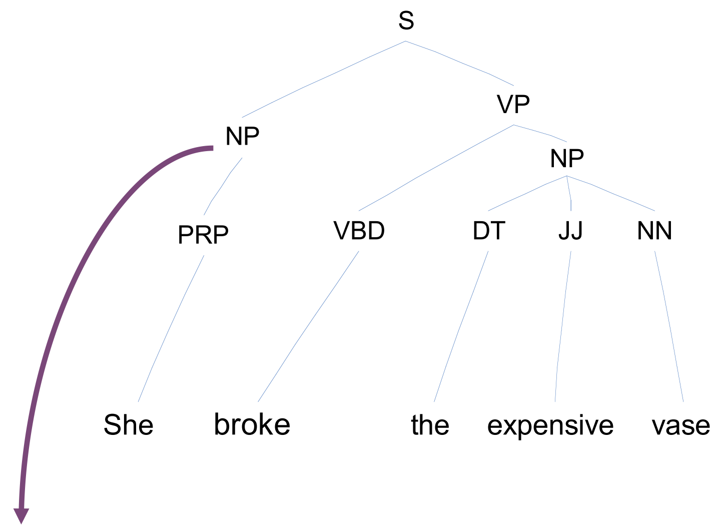
- Stage 1: Filter out constituents that are clearly not semantic arguments to the predicate in question (saves time)
- Stage 2: Classify the candidates derived from the first stage as either semantic arguments or non-arguments.
- Stage 3: Run a multi-category classifier to classify the constituents that are labeled as arguments into one of the classes plus NULL.

Gildea & Jurafsky (2002) Features



- Key early work
  - Future systems use these features as a baseline

- Constituent Independent
  - Target predicate (lemma)
  - Voice
  - Subcategorization
- Constituent Specific
  - Path
  - Position (*left, right*)
  - Phrase Type
  - Governing Category (*S* or *VP*)
  - Head Word

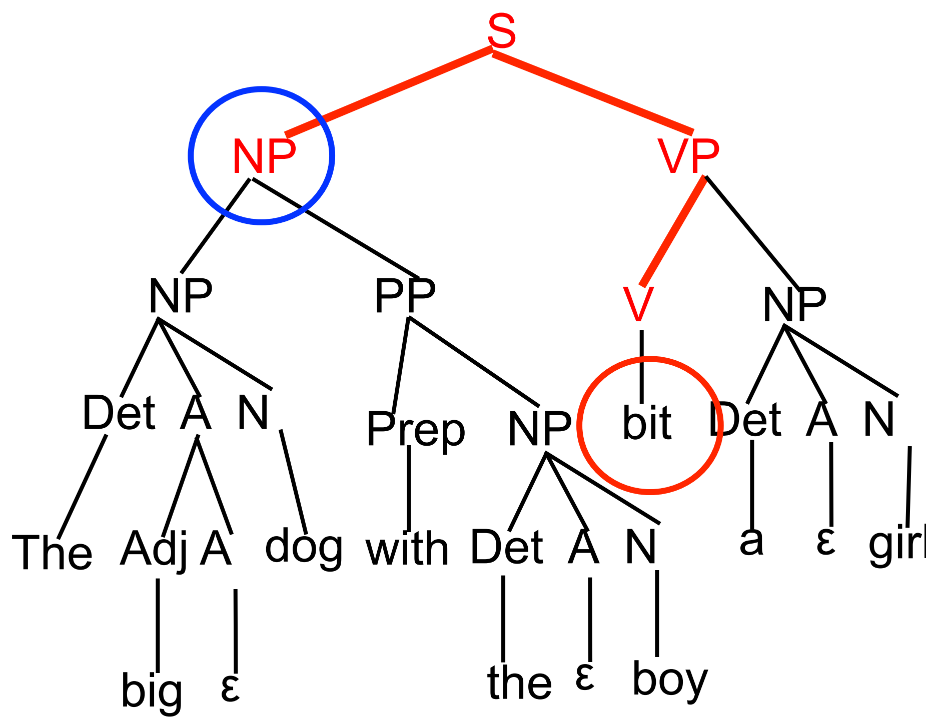


Target	*broke*
Voice	*active*
Subcategorization	*VP→VBD NP*
Path	*VBD↑VP↑S↓NP*
Position	*left*
Phrase Type	*NP*
Gov Cat	*S*
Head Word	*She*

+ Parse Tree Path Feature: Example 1

Path Feature Value:

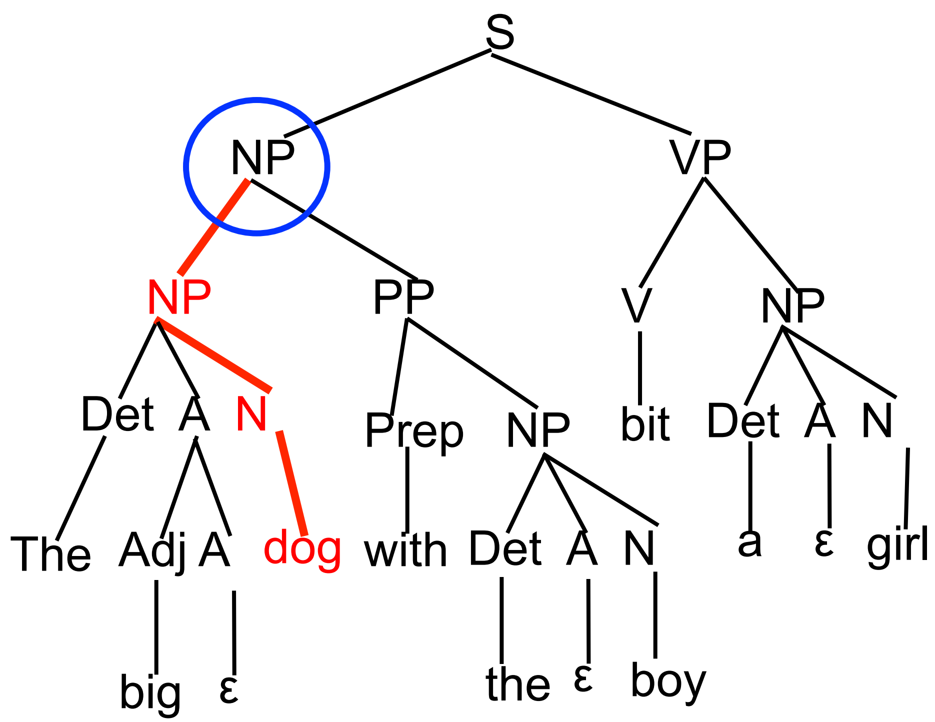
V ↑ VP ↑ S ↓ NP



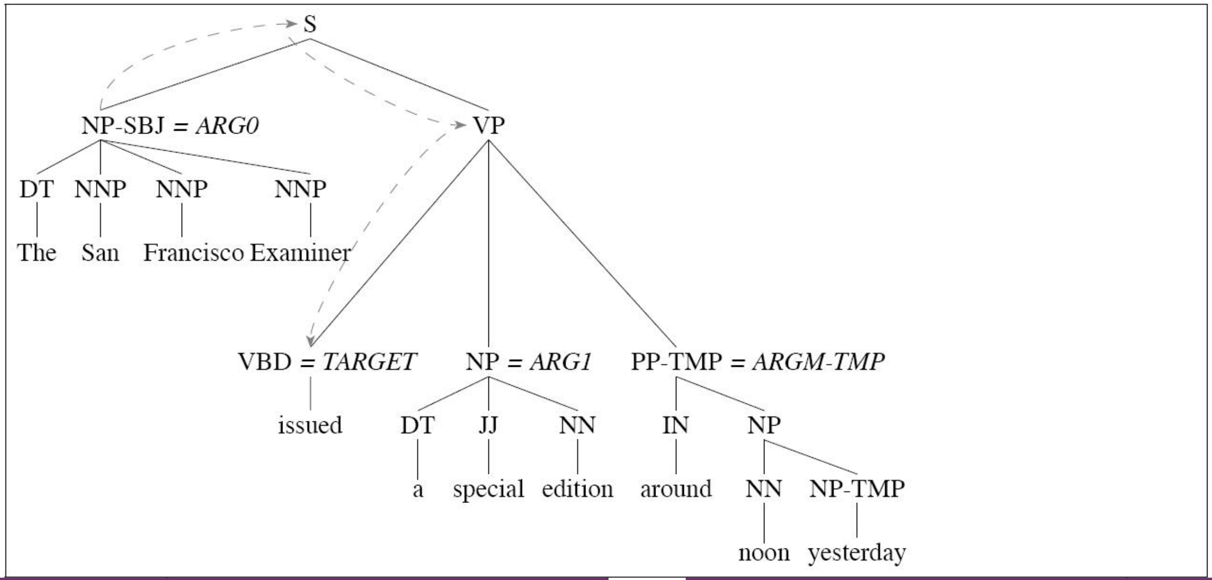
+ Head Word Feature Example

- There are standard syntactic rules for determining which word in a phrase is the **head**.

Head Word:
dog



Another example



Target	*issued*
Voice	*active*
Subcategorization	*VP→VBD NP PP*
Path	*VBD↑VP↑S↓NP*
Position	*left*
Phrase Type	*NP*
Gov Cat	*S*
Head Word	*Examiner*

Target	*issued*
Voice	*active*
Subcategorization	*VP→VBD NP PP*
Path	*VBD↑VP↓NP*
Position	*right*
Phrase Type	*NP*
Gov Cat	*VP*
Head Word	*edition*

+ Summary "Standard" features

- Predicate The predicate itself.
- Path The minimal path from the constituent being classified to the predicate.
- Phrase Type The syntactic category (NP, PP, etc.) of the constituent being classified.
- Position The relative position of the constituent being classified with regard to the predicate (before or after)
- Voice Whether the predicate is active or passive.
- Head Word The head word of the constituent being classified.
- Sub-categorization The phrase structure rule expanding the parent of the predicate.

+ Argument Identification

- A subset of features and their combination contribute most to argument identification
  - path,
  - head word, head word part-of-speech,
  - predicate - phrase type combination,
  - predicate- head word combination,
  - distance between constituent and predicate, with the predicate specified.

+ Argument identification

- Some features do not help discriminate argument identification
  - path: Can’t distinguish between sisters
    - Direct object & indirect object not distinct
  - Subcategorization: Shared by all of the arguments
  - Voice: Same for all args, mabey combine with arg/label
  - phrase type: Does help but would be stronger if pared with the predicate
  - head word: Also should be pared with predicate

+ New features for Argument Identification

- Syntactic frame: varies with the constituent being classified to complement the path and subcat features
- Lexicalized constituent type: combination of the predicate lemma and the phrase type, rather than the phrase type itself, e.g. give np.
- Lexicalized head : predicate lemma and the head word combination as a feature, e.g. give states.
- Voice position combination: voice position combination as a feature, e.g. passive before.
- Head of PP: parent If the parent of the current constituent is a PP, then the head of this PP, the preposition is also used as a feature.

+ Performance per feature

Features	Accuracy	Gold(f)
Baseline	88.09	82.89
Syntactic frame	89.82	84.64
Pred-Head	88.69	83.77
Pred-POS	89.12	83.81
Voice position	88.44	82.57
PP parent	89.53	84.34
First word	88.60	83.01
Last word	88.64	83.51
Left sister	89.20	83.74
all	92.95	88.51

+ What is Feature selection ?

- Feature selection:
  Problem of selecting some subset of a learning algorithm's input variables upon which it should focus attention, while ignoring the rest (DIMENSIONALITY REDUCTION)

- Humans/animals do this constantly

+ Nomenclature

- **Univariate method**: considers one variable (feature) at a time.
- **Multivariate method:** considers subsets of variables (features) together.
- **Filter method:** ranks features or feature subsets independently of the predictor (classifier).
- **Wrapper method:** uses a classifier to assess features or feature subsets.

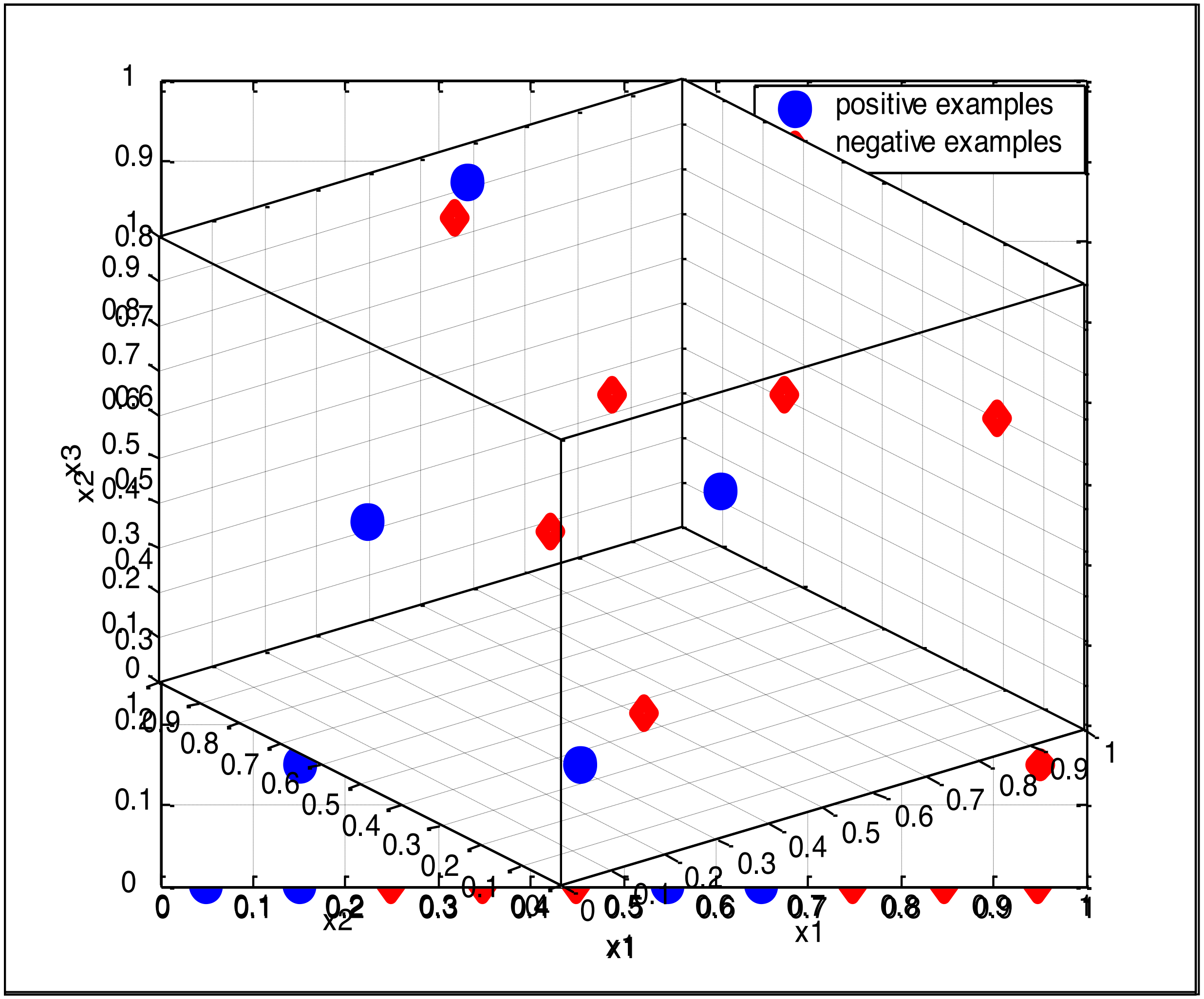
+ Feature Selection in ML ?

- Why even think about Feature Selection in ML?
  - The information about the target class is inherent in the variables!
  - Naive theoretical view:
    More features
    => More information
    => More discrimination power.
  - In practice:
    many reasons why this is not the case!
  - Also:
    Optimization is (usually) good, so why not try to optimize the input-coding ?

+ Feature Selection in ML

- Many explored domains have hundreds to tens of thousands of variables/features with many irrelevant and redundant ones
  - In domains with many features the underlying probability distribution can be very complex and very hard to estimate (e.g. dependencies between variables)
- Irrelevant and redundant features can confuse learners
- Limited training data
- Limited computational resources
- Curse of dimensionality

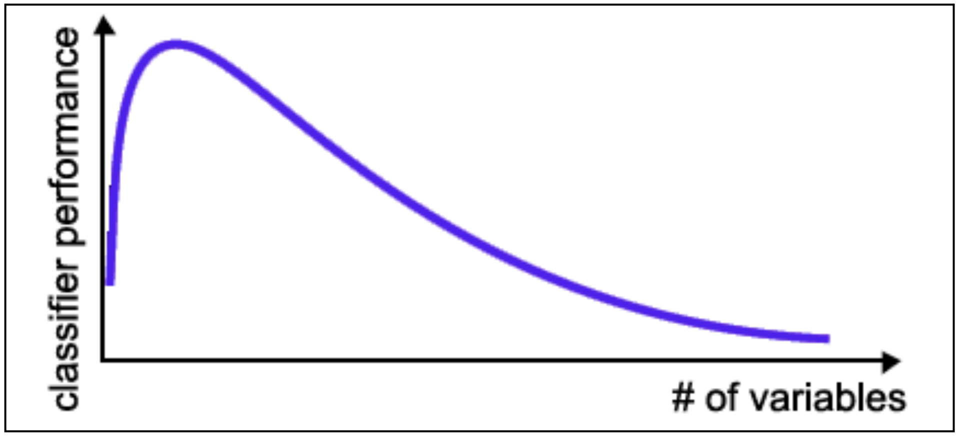
Curse of dimensionality



Curse of dimensionality

- The required number of samples (to achieve the same accuracy) grows exponentially with the number of variables!
- In practice: number of training examples is fixed!

  => the classifier's performance usually will degrade for a large number of features!



In many cases the information that is lost by discarding variables is made up for by a more accurate mapping/sampling in the lower-dimensional space !

+ Example for ML-Problem

- Gene selection from microarray data
  - Variables:
    - gene expression coefficients corresponding to the amount of mRNA in a patient ‘s sample (e.g. tissue biopsy)
  - Task: Separate healthy patients from cancer patients
  - Usually there are only about 100 examples (patients) available for training and testing (!!!)
  - Number of variables in the raw data: 6.000 – 60.000
  - Does this work ?

+ Example for ML-Problem

- Text-Categorization
  - Documents are represented by a vector of dimension the size of the vocabulary containing word frequency counts
  - Vocabulary ~ 15,000 words (i.e. each document is represented by a 15,000-dimensional vector)
  - Typical tasks:
    - Automatic sorting of documents into web-directories
    - Detection of spam-email

+ Motivation

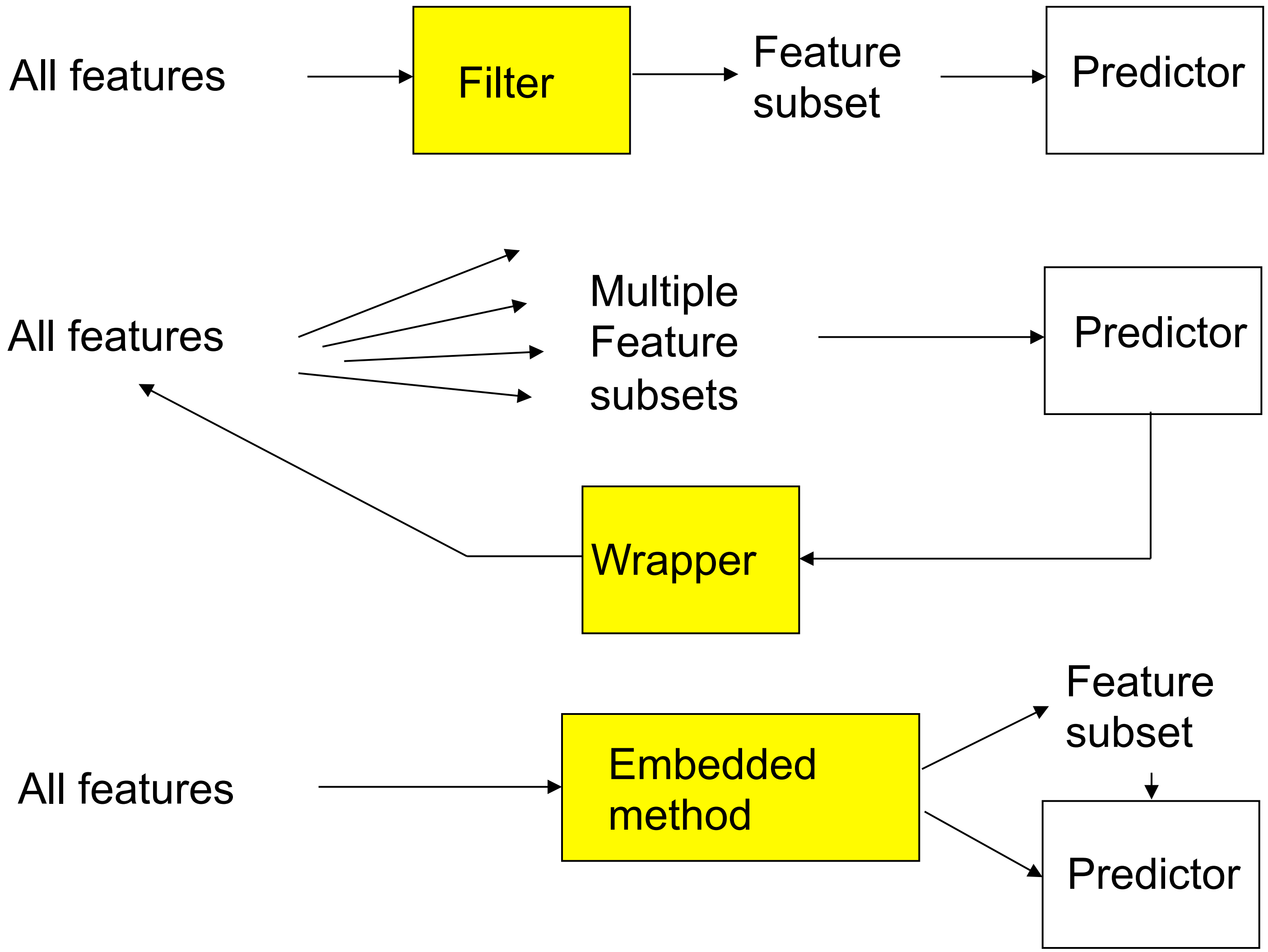
- Especially when dealing with a large number of variables there is a need for dimensionality reduction

- Feature Selection can significantly improve a learning algorithm’s performance

+ Approaches

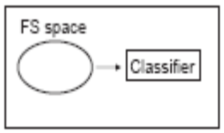
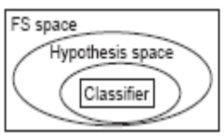
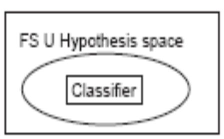
- Wrapper
  - feature selection takes into account the contribution to the performance of a given type of classifier
  - Train a classifier with a subset of the features and look at the results
- Filter
  - feature selection is based on an evaluation criterion for quantifying how well feature (subsets) discriminate the two classes
  - Use a measure such as mutual information or pointwise mutual information to rank features and a cut off point using techniques such as cross validation
  - Can be a preprocess for wrapper methods
- Embedded
  - feature selection is part of the training procedure of a classifier (e.g. decision trees)
  - Support Vector Machines using Recursive Features Elimination repeatedly constructs a model and removes features with low weights
  - Computational complexity is midway between wrapper and filter

+ Filters, Wrappers, and Embedded methods



+ Feature Selection techniques in a nutshell

Table 1. A taxonomy of feature selection techniques. For each feature selection type, we highlight a set of characteristics which can guide the choice for a technique suited to the goals and resources of practitioners in the field.

	Model search		Advantages	Disadvantages	Examples
Filter		Univariate	Fast Scalable Independent of the classifier	Ignores feature dependencies Ignores interaction with the classifier	Chi-square Euclidean distance t-test Information gain, Gain ratio [6]
		Multivariate	Models feature dependencies Independent of the classifier Better computational complexity than wrapper methods	Slower than univariate techniques Less scalable than univariate techniques Ignores interaction with the classifier	Correlation based feature selection (CFS) [45] Markov blanket filter (MBF) [62] Fast correlation based feature selection (FCBF) [136]
Wrapper		Deterministic	Simple Interacts with the classifier Models feature dependencies Less computationally intensive than randomized methods	Risk of over fitting More prone than randomized algorithms to getting stuck in a local optimum (greedy search) Classifier dependent selection	Sequential forward selection (SFS) [60] Sequential backward elimination (SBE) [60] Plus q take-away r [33] Beam search [106]
		Randomized	Less prone to local optima Interacts with the classifier Models feature dependencies	Computationally intensive Classifier dependent selection Higher risk of overfitting than deterministic algorithms	Simulated annealing Randomized hill climbing [110] Genetic algorithms [50] Estimation of distribution algorithms [52]
Embedded			Interacts with the classifier Better computational complexity than wrapper methods Models feature dependencies	Classifier dependent selection	Decision trees Weighted naive Bayes [28] Feature selection using the weight vector of SVM [44, 125]

Saeys Y, Inza I, Larrañaga P. A review of feature selection techniques in bioinformatics Bioinformatics. 2007 Oct 1;23(19):2507-1

Creating attribute-value table

	f_1	f_2	…	f_K	y
x_1					
x_2					
…					

- Choose features:
  - Define feature templates
  - Instantiate the feature templates
  - Dimensionality reduction: feature selection
- Feature weighting
  - The weight for f_k: the whole column
  - The weight for f_k in d_i: a cell

+ An example: text classification task

- Define feature templates:
  - One template only: word
- Instantiate the feature templates
  - All the words appeared in the training (and test) data
- Dimensionality reduction: feature selection
  - Remove stop words
- Feature weighting
  - Feature value: term frequency (tf), or tf-idf

Dimensionality reduction (DR)

- What is DR?
  - Given a feature set r, create a new set r', s.t.
    - r' is much smaller than r, and
    - the classification performance does not suffer too much.

- Why DR?
  - ML algorithms do not scale well.
  - DR can reduce overfitting.

Term selection vs. extraction

- Term selection: r’ is a subset of r
  - Wrapping methods: score terms by training and evaluating classifiers.

    ➔ expensive and classifier-dependent
  - Filtering methods
- Term extraction: terms in r’ are obtained by combinations or transformation of r terms.
  - Term clustering:
  - Latent semantic indexing (LSI)

+ Term selection by filtering

- Main idea: scoring terms according to predetermined numerical functions that measure the “importance” of the terms.
- It is fast and classifier-independent.
- Scoring functions:
  - Information Gain
  - Mutual information
  - chi square
  - …

+ Calculating basic distributions over categories and tokens

	c_i	$\overline{c}_i$
t_k	A	B
$\overline{t}_i$	C	D

- A - Number of documents in CATEGORY C, containing WORD t.
- B - Number of documents **not** in CATEGORY C, containing WORD t.
- C - Number of documents in CATEGORY C, **not** containing WORD t.
- D - Number of documents **not** in CATEGORY C, **not** containing WORD t.

+ Types of DR

- r is the original feature set, r' is the one after DR.
- Local DR vs. Global DR
  - Global DR: r' is the same for every category
  - Local DR: a different r' for each category
- Term extraction vs. term selection

+ Calculating basic distributions over categories and tokens

	c_i	$\overline{c_i}$	T
t_k	a	b	f
$\overline{t_i}$	c	d	h
	e	g	N

- $P(t_k, c_i) = a/N$
- $P(t_k) = (a + b) /N \;\; = f / N$
- $P(c_i) = (b + d) /N \;\;\; = g / N$
- $N = a + b + c + d$

+ Term selection functions

- Intuition: for a category c_i , the most valuable terms are those that are distributed most <u>differently</u> in the sets of possible and negative examples of c_i.

+ Term selection functions

- Intuition: for a category c_i , the most valuable terms are those that are distributed most <u>differently</u> in the sets of possible and negative examples of c_i.
- Document frequency: The number of documents in which t_k appears.
- Mutual Information

$$\mathrm{MI}(t_k, c_i) = \log \frac{P(t_k, c_i)}{P(c_i)P(t_k)}$$

- Information Gain

$$IG(t_k, c_i) = P(t_k, c_i)\log \frac{P(t_k, c_i)}{P(c_i)P(t_k)} + P(\overline{t_k}, c_i)\log \frac{P(t_k, \overline{c_i})}{P(c_i)P(\overline{t_k})}$$

+ Information gain

- IG(Y|X): We must transmit Y. How many bits on average would it save us if both ends of the line knew X?

- Definition:

$$IG(Y, X) = H(Y) - H(Y|X)$$

More term selection functions**

GSS coefficient: Galavotti-Sebastiani-Simi

$$GSS(t_k, c_i) = P(t_k, c_i)P(\bar{t_k}, \bar{c_i}) - P(t_k, \bar{c_i})P(\bar{t_k}, c_i)$$

Ng-Goh-Low-Leong

NGL coefficient: N is the total number of docs

$$NGL(t_k, c_i) = \frac{\sqrt{N}\ GSS(t_k, c_i)}{\sqrt{P(t_k)P(\bar{t_k})P(c_i)P(\bar{c_i})}}$$

Chi-square: (one of the definitions)

$$\chi^2(t_k, c_i) = NGL(t_k, c_i)^2 = \frac{(ad-bc)^2 N}{(a+b)(a+c)(b+d)(c+d)}$$

+ More term selection functions**

Relevancy score:

$$RS(t_k, c_i) = log\frac{P(t_k|c_i)+d}{P(\bar{t_k}|\bar{c_i})+d}$$

Odds Ratio:

$$OR(t_k, c_i) = \frac{P(t_k|c_i)P(\bar{t_k}|\bar{c_i})}{P(\bar{t_k}|c_i)P(t_k|\bar{c_i})}$$

+ Global DR

- For local DR, calculate $f(t_k, c_i)$.

- For global DR, calculate one of the following:

$$\text{Sum: } f_{sum}(t_k) = \sum_{i=1}^{|C|} f(t_k, c_i)$$

$$\text{Average: } f_{avg}(t_k) = \sum_{i=1}^{|C|} f(t_k, c_i) P(c_i)$$

$$\text{Max: } f_{max}(t_k) = \max_{i=1}^{|C|} f(t_k, c_i)$$

|C| is the number of classes

+ Alternative feature values

- Binary features: 0 or 1.
- Term frequency (TF): the number of times that t_k appears in d_i.
- Inversed document frequency (IDF): log |D| /d_k, where d_k is the number of documents that contain t_k.
- TFIDF = TF * IDF
- Normalized TFIDF:

$$w_{ik} = \frac{tfidf(d_i,t_k)}{Z}$$

+ Feature weights

- Feature weight 2 {0,1}: same as DR

- Feature weight 2 R: iterative approach:
  - Ex: MaxEnt

➔ Feature selection is a special case of feature weighting.

+ Summary so far

- Curse of dimensionality ➔ dimensionality reduction (DR)

- DR:
  - Term extraction
  - Term selection
    - Wrapping method
    - <u>Filtering method</u>: different functions

+ Summary (cont)

- Functions:
  - Document frequency
  - Mutual information
  - Information gain
  - Gain ratio
  - Chi square
  - …