



machine learning for IR



Text and Machine Learning

- Information Retrieval
- Library and Information Science
Artificial Intelligence
- Natural Language Processing
- Database Management



What is Machine Learning?

- A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E . [Mitchell '97]
- T : Classifying Text to some category
- P : Accuracy of Classification
- E : A training set

machine learning

- Given such a dataset onemight want to:
 - Learn to put Instances into predefined classes (classification)
 - Learn relationships between attributes (association learning)
 - Groups similar instances together (clustering)

- A fictional dataset

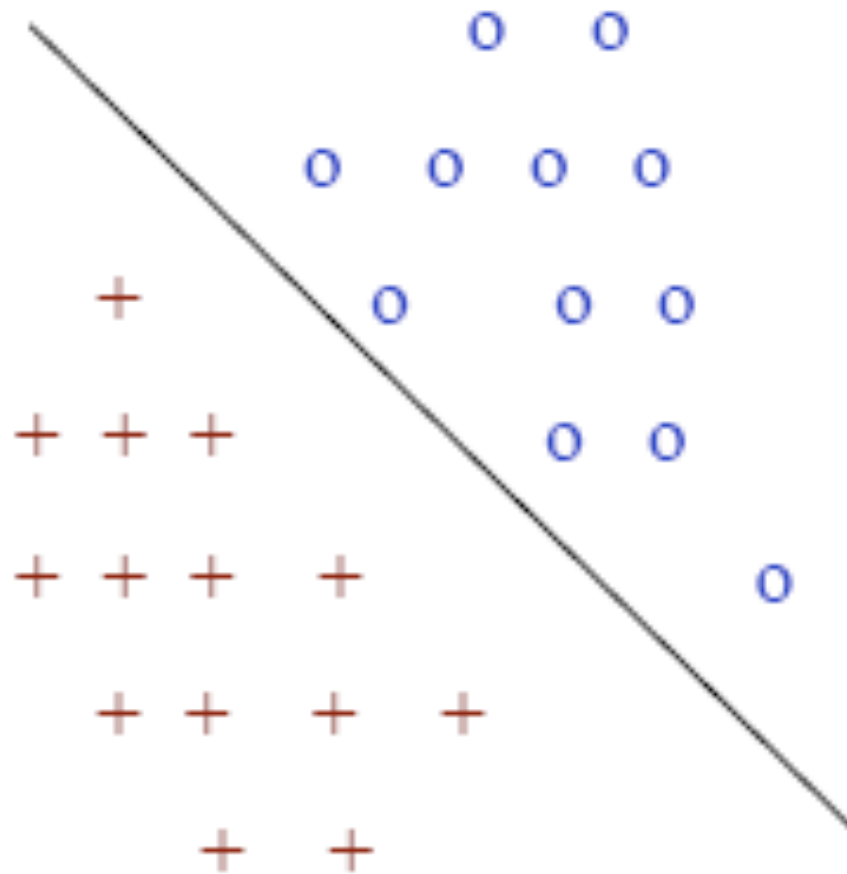
Name	Age	Sex	...	Risk
Tom	32	M	...	Y
Mary	54	F	...	N
John	13	M	...	?
Kim	10	F	...	?
...

pattern classification

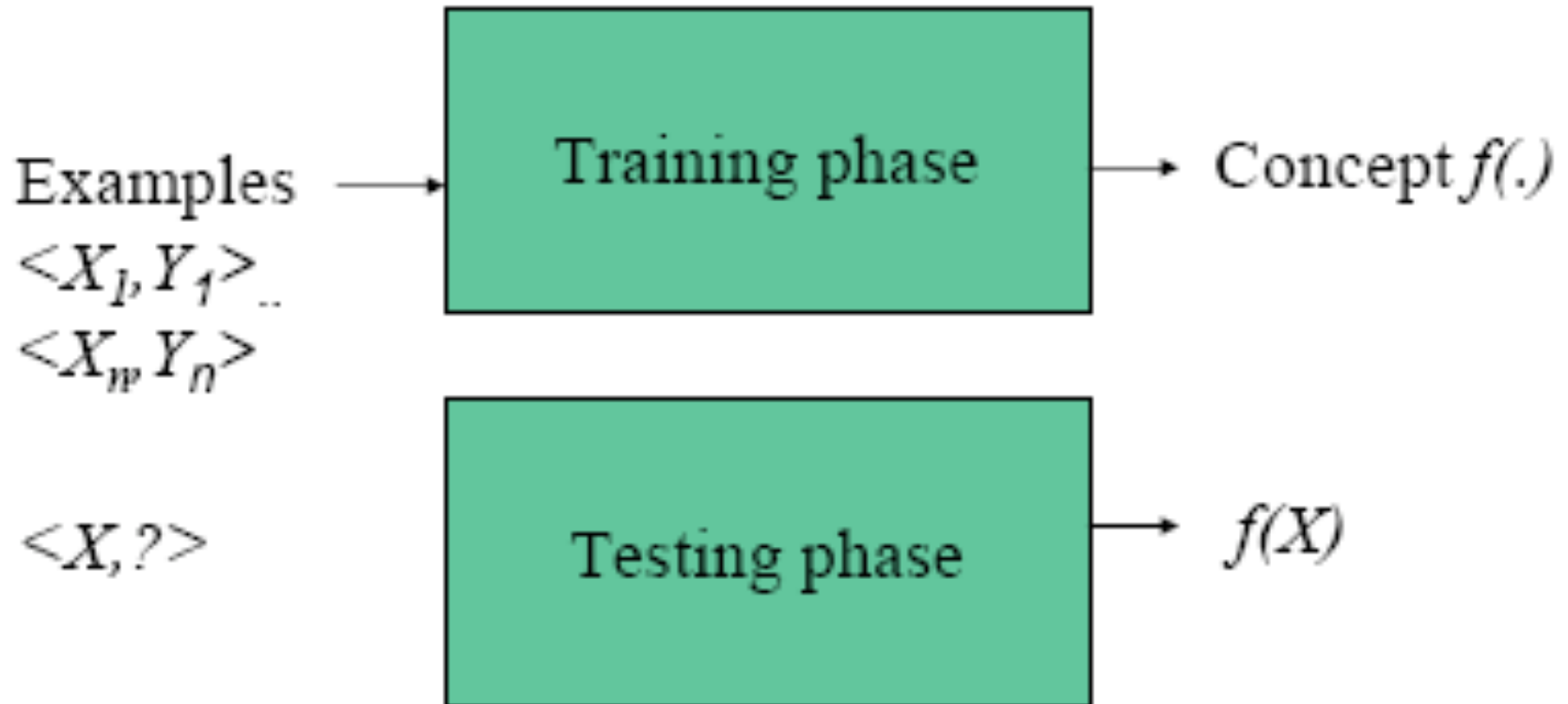
- Definitions:
 - Instance: Single example in the dataset (X_i)
 - Attribute: An aspect of an instance x_j
 - Value: Value that an attribute can take
 - $X=(X_1 \dots X_n)$, a set of d-dimensional vectors (the data)
 - $X_i = x_{1,i} \dots x_{m,i}$
 - $Y=Y_1 \dots Y_m$, a set of output classes
 - Concept - The thing to be learned

Name	Age	Sex	...	Risk
Tom	32	M	...	Y
Mary	54	F	...	N
John	13	M	...	?
Kim	10	F	...	?
...

example concept



training and testing

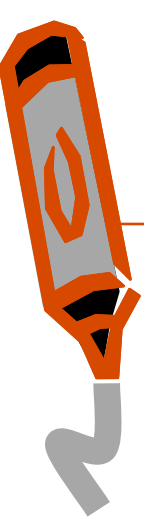




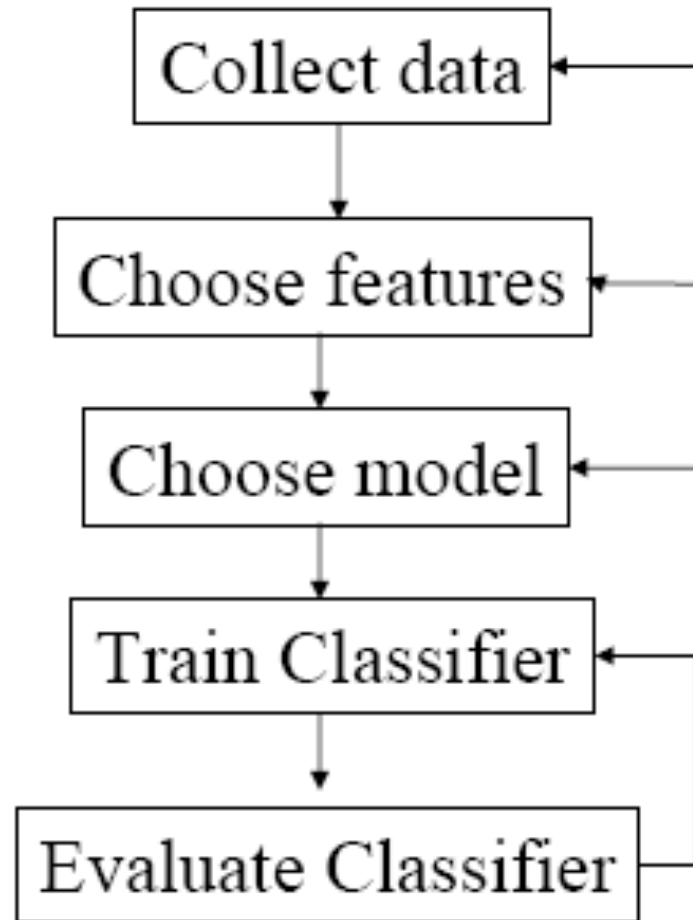
- Document Classification
- Standard datasets:
 - Reuters: Reuters news articles in categories like earnings, acquisitions etc
 - Newsgroups: Newsgroups pages: Predict the newsgroup (comp.graphics, comp.os.mswindows.misc, rec.sport.baseball, rec.sport.hockey etc)

Docs \ Features	w1	...	wn	...	Class
1					
2					
3					
4					
...		

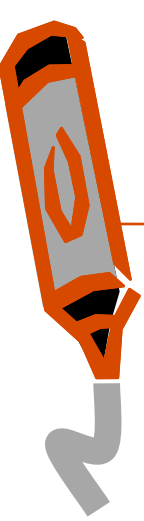
classification



Cross Validation



Supervised Learning



- Supervised learning
 - learning algorithm is provided with a set of inputs for the algorithm along with the corresponding correct outputs,
 - learning involves the algorithm comparing its current actual output with the correct or target outputs, so that it knows what its error is, and modify things accordingly.
- Unsupervised Learning
 - Example - regression, clustering



models

- Discriminative Models:

$$x \rightarrow g(x)$$

- Generative models:

$$x \rightarrow P(x|C)$$

$$P(C|x) \propto P(x|C)P(C)$$

$$g(x) = \frac{P(C|x)}{P(\bar{C}|x)}$$



naive Bayes

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)}$$

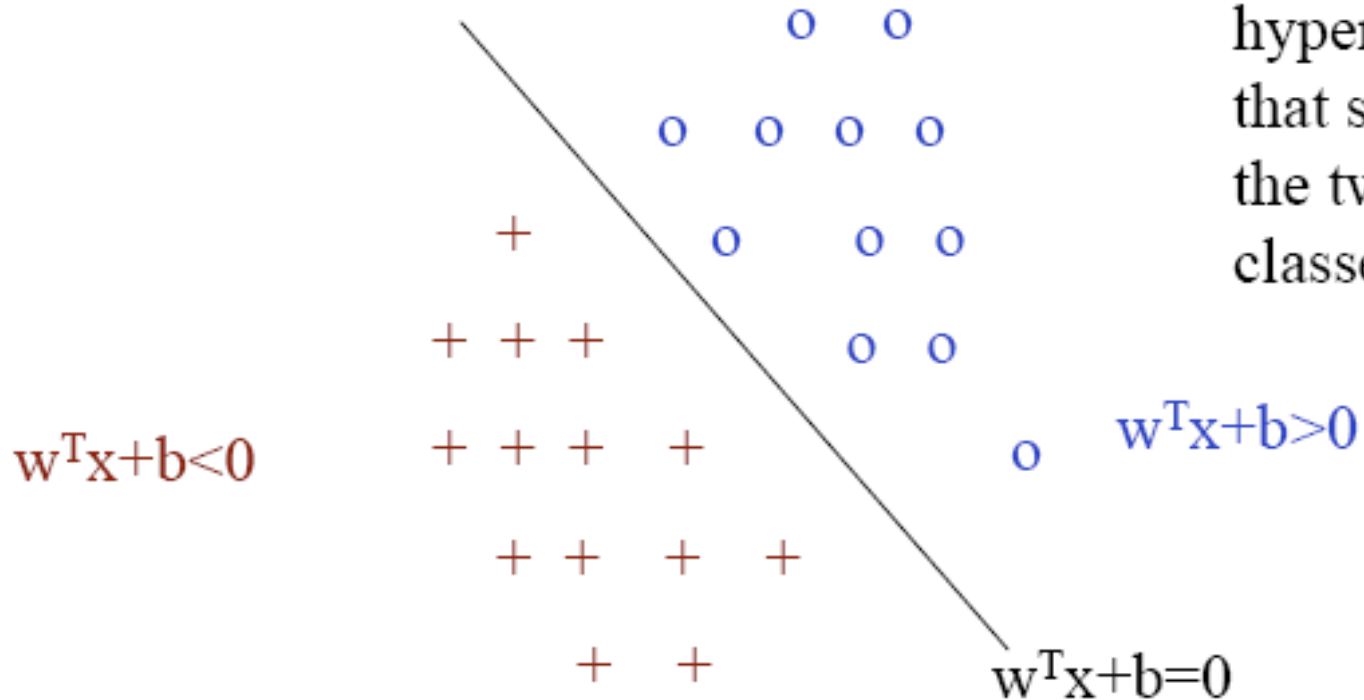
$$P(X|C) = \prod_i^{|V|} p(x_i|C)$$

- If $P(C|X) > P(\bar{C}|X)$ then assign X to C
 - Intuitive. Also corresponds to the action where Bayes Risk is minimum
- Example of Generative Model
- Probabilities are Max likelihood with some form of smoothing

support vector machines

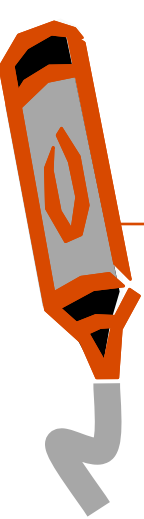


Find the best hyper-plane that separates the two classes



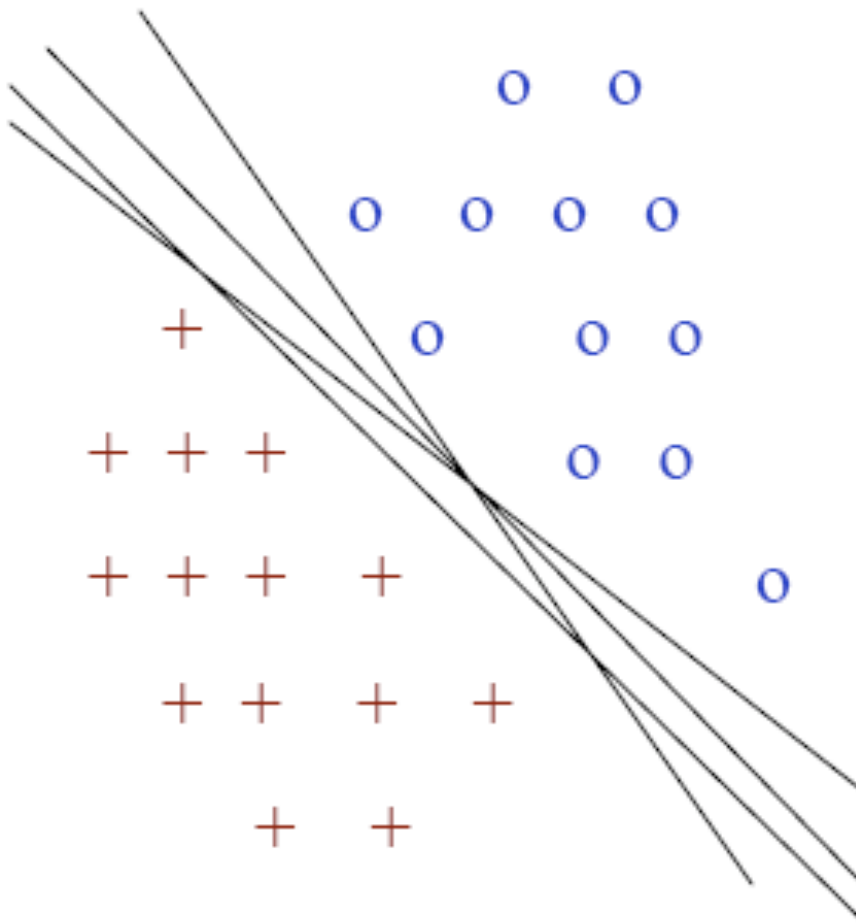
Example of a Generative Model

$$f(x) = \text{sgn}(w^T x + b)$$

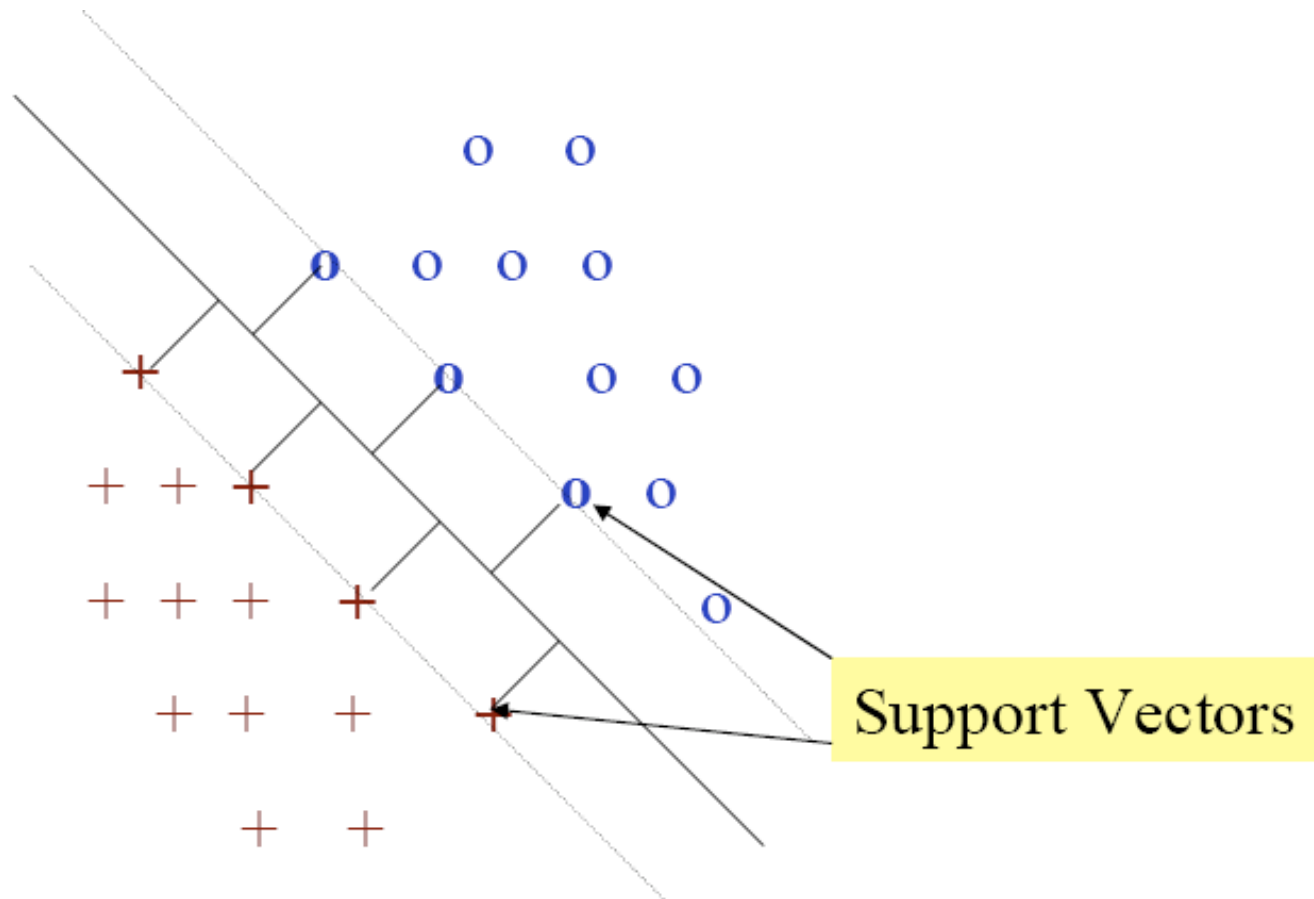


support vector machines

But what is
the best
hyper-plane?



support vector machines



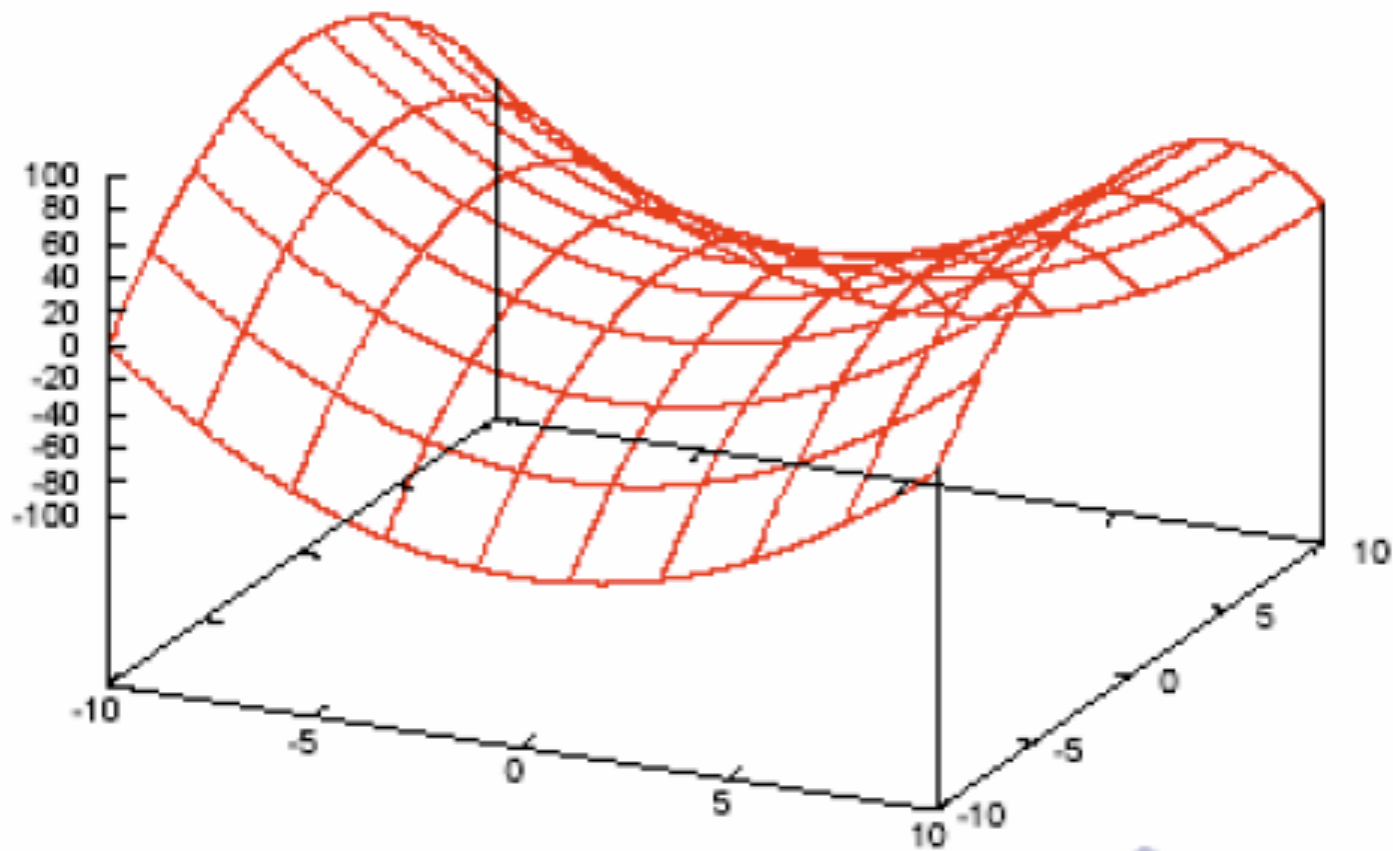


support vector machines

- optimization problem

$$(w^*, b^*) = \operatorname{argmax}_{(w, b)} \min_{(X_i \in X)} Y_i (w^T X_i + b)$$

Lagrange optimization



svm

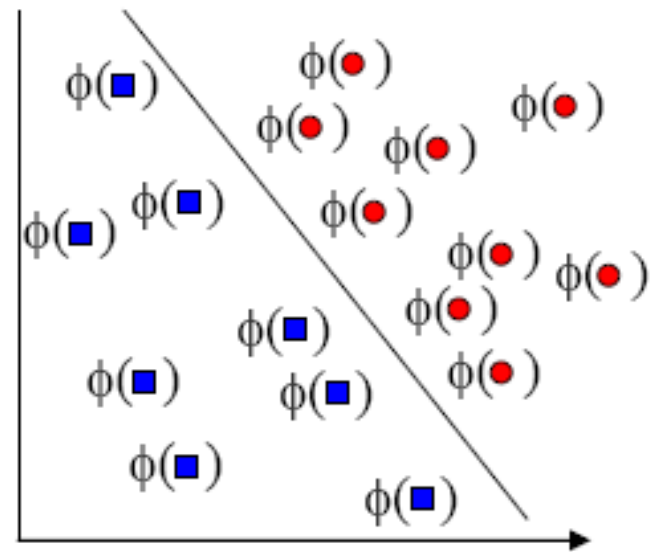
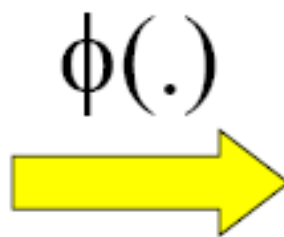
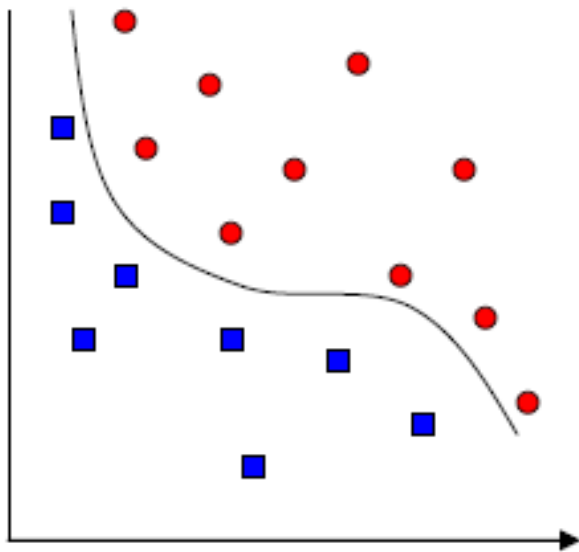
- The solution is of the form

$$f(x) = \text{sgn}\left(\sum_{i \in SV} \alpha_i y_i x_i^T x + b^*\right)$$

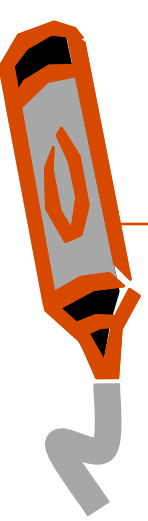
- Support vectors are the only important data points in the training set
- Summation over number of support vectors

the kernel trick

$$K(x, y) = \phi(x)^T \phi(y)$$



IR as a Classification Problem



- Binary Classification and
- Compare with Language Modeling Framework

Probabilistic IR models as classifiers

- BIR model : A generative classifier
 - Features are binary representing the presence or absence of each word in the vocabulary
 - Uses a multiple-Bernoulli model to model the class-conditional

$$\begin{aligned}\log \frac{P(R | \mathbf{D})}{P(\bar{R} | \mathbf{D})} &= \log \frac{P(\mathbf{D} | R)P(R)}{P(\mathbf{D} | \bar{R})P(\bar{R})} \\ &= \log \left(\prod_{i:x_i=1} \frac{P(x_i = 1 | R)}{P(x_i = 1 | \bar{R})} \prod_{i:x_i=0} \frac{P(x_i = 0 | R)}{P(x_i = 0 | \bar{R})} \right)\end{aligned}$$

Probabilistic IR models as classifiers

- Language models

- Appear to have abandoned the notion of IR as a binary classification problem: There is no reference to the class variable R !
- However, if we imagine each document as a unique class, language models can be considered generative!
- Language models rank the classes (documents) for each instance (query)!

Case for Discriminative models for IR



- Theoretical considerations
 - “One should solve the (classification) problem directly and never solve a more general problem (class-conditional) as an intermediate step” [Vapnik, 1998]
 - Discriminative models tend to have a lower asymptotic error as the training set size is increased [Ng and Jordan, NIPS 2002]

Case for Discriminative models for IR

- Modeling assumptions
 - Term conditional independence assumptions in LM not strictly valid
 - Multinomial distribution fails to model burstiness of terms [Teevan and Karger, SIGIR 2003]
 - Discriminative models make very few assumptions and let the data speak for itself!


Case for Discriminative models for IR

- Case for Discriminative models for IR
- Expressiveness : advanced features
 - Proximity of query terms
 - Ordering of terms
 - Presence or absence of terms
- Hard to include such features in LMs
- Discriminative models can handle arbitrary features

Case for Discriminative models for IR

- Learning arbitrary features
 - Multiple representations of documents
 - E.g.: abstract, title, anchor text, document content
 - Query-independent features
 - E.g.: Page Rank
 - User preferences
- Language models permit both but feature weights (typically) determined empirically
- Discriminative models can learn all such features automatically

IR vs. Text Classification


- 
- IR not same as text classification!
 - IR is much harder: training data is very sparse
 - Dynamic vs. static classes:
Distribution of words in the relevant class is query-specific
 - training on words as features will not help
 - Features based on query-based statistics of documents instead

Unbalanced data




- Non-relevant class is represented by much larger number of training examples than the relevant class
- Discriminative classifiers trained on unbalanced data result in trivial classifiers
- Methods used to overcoming unbalanced data problem:
 - Oversampling minority class
 - Undersampling majority class
 - Adjusting misclassification cost of one of the classes

Ad-hoc Retrieval

- 
- Task of retrieving a ranked list of relevant documents for a given free-text query
 - 4 different TREC collections used in the experiments: each collection has a set of train and test queries and relevance judgments
 - SVM and LM
 - The models trained on each collection and tested on all 4 collections: in total we have 16 runs
 - Documents and queries are pre-processed using a stop-word list and the K-stemmer

Ad-hoc Retrieval

- 
- Used title queries in all experiments
 - Dirichlet smoothing is used in LM runs: training consists of finding the best value of Dirichlet parameter
 - SVMs: linear kernels proved the best
 - Discriminative models trained using all relevant examples and randomly sampled non-relevant examples
 - Lemur for LMs, SVM-light for SVMs

Ad-hoc Retrieval

Features used in the discriminative models

$$1. \sum_{i=1}^n \log(c(q_i, D))$$

$$2. \sum_{i=1}^n \log\left(1 + \frac{c(q_i, D)}{|D|}\right)$$

$$3. \sum_{i:c(q_i, D) > 0} \log(idf(q_i))$$

$$4. \sum_{i:q_i > 0} \log \frac{|C|}{c(q_i, C)}$$

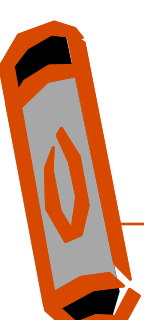
$$5. \sum_{i=1}^n \log\left(1 + \frac{|D|}{c(q_i, D)} idf(q_i)\right)$$

$$6. \sum_{i=1}^n \log\left(1 + \frac{|C|}{c(q_i, C)} \frac{c(q_i, D)}{|D|}\right)$$



Out-Of-Vocab problem

- Words in test queries are mostly to have occurred in training queries.
- However, features are based not on words but on the term statistics.



adhoc retrieval



Train	Test	Disk 1-2 (151-200)	Disk3 (101-150)	Disks 4-5 (401-450)	WT2G (426-450)
Disk1-2 (101-150)	LM	0.2561 (6.75e-3)	0.1842	0.2377 (0.80)	0.2665 (0.61)
	SVM	0.2145	0.1877 (0.3)	0.2356	0.2598
Disk3 (51-100)	LM	0.2605 (1.08e-4)	0.1785 (0.11)	0.2503 (0.21)	0.2666
	SVM	0.2064	0.1728	0.2432	0.2750 (0.55)
Disk4-5 (301-350)	LM	0.2592 (1.75e-4)	0.1773 (7.9e-3)	0.2516 (0.036)	0.2656
	SVM	0.2078	0.1646	0.2355	0.2675 (0.89)
WT2G (401-425)	LM	0.2524 (4.6e-3)	0.1838 (0.08)	0.2335	0.2639
	SVM	0.2199	0.1744	0.2487 (0.046)	0.2798 (0.037)



ad-hoc retrieval

- Conclusions
 - LMs, despite some inaccurate assumptions are quite robust!
 - class conditional models using a fixed distribution are relatively impervious to noise in training data
 - Simplicity helps in good generalization
 - Why use SVMs then?
 - Strength of SVMs: ability to learn relative importance of arbitrary features automatically



home page finding

- Task of retrieving the relevant document as high in the ranked list as possible.
 - Corpus is WT10G, a 10GB web collection.
 - 50 Queries for Training, 50 for development and 145 for testing
 - Evaluation
 - Mean Reciprocal Rank (MRR)
 - Success rate
 - Failure rate

home page finding

- Features used in discriminative models
 - Query-dependent features:
 - Document content
 - Anchor text
 - Title
 - Query-independent features
 - Link factor
- URL-depth: reciprocal of number of branches in the depth of path of the document
$$\log\left(1 + \frac{\text{num} - \text{links}(D)}{\text{Avg} - \text{num} - \text{links}}\right)$$



home page finding

Results on the development set

SVM features	MRR	Success %	Failure %
Content + Anchor	0.54	73.0	5.2
Content + Anchor + Title	0.61	85.7	10.2
Content + Anchor + Title + URL	0.61	85.7	10.2
Content + Anchor + Title + URL + link	0.61	85.7	10.2
LM baseline	0.35	52.0	10.0
SVM baseline	0.33	53.06	12.24



home page finding

Results on test set

- Used all query-dependent and query-independent features

Model	MRR	Success %	Failure %
Full-featured SVM	0.52	77.93	11.03
LM baseline	0.35	57.93	15.86
SVM Baseline	0.28	52.41	17.90



Different Learning Paradigms

- Inductive Learning – what you just saw
 - Learn from solved examples in a book . In-class closed book exam
- Active Learning
 - Only unsolved problems. Can ask an expert a few questions. In-class closed book exam
- Semi supervised learning
 - Book examples, back of the book questions. In-class closed book exam
- Transductive Learning.
 - Book examples. Take home exam.



Active Learning

- In Active Learning the learner can ask an expert the labels of some of the unlabeled instances in order to improve classification accuracy.
- The objective is to ask the expert as few questions as possible.
- Uncertainty sampling is one way of Active Learning



Active Learning

- Query by Committee [Freund, Sueng et al]
 - They prove theoretically that if a 2 member committee can achieve information gain with +ve lower bound then error decreases exponentially in the number of queries
- Uncertainty Sampling [Lewis and Gale]
 - Query on those instances that the Naïve Bayes classifier is most uncertain about ($p(Y|X) \sim 0.5$)
- Optimize on expected future error [Roy, McCallum]
- Active Learning with Support Vector Machines [Tong, Koller]
 - Pick a sample such that the knowledge of the label reduces the version space in half.

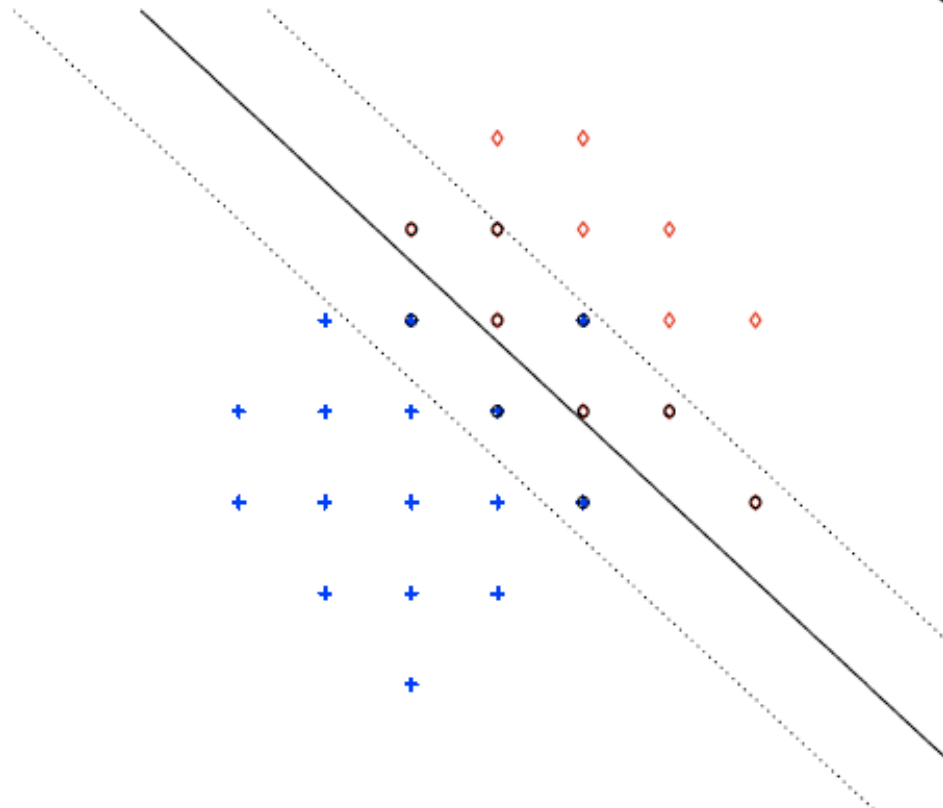


Active Learning with a Naive Bayes Classifier

- Remember the Naïve Bayes Classifier
- The simplest way of uncertainty sampling is to query the user on instances with as close to 0.5 as possible.

$$\frac{P(C|D)}{P(\bar{C}|D)} = \frac{P(C)}{P(\bar{C})} \times \frac{P(D|C)}{P(D|\bar{C})}$$

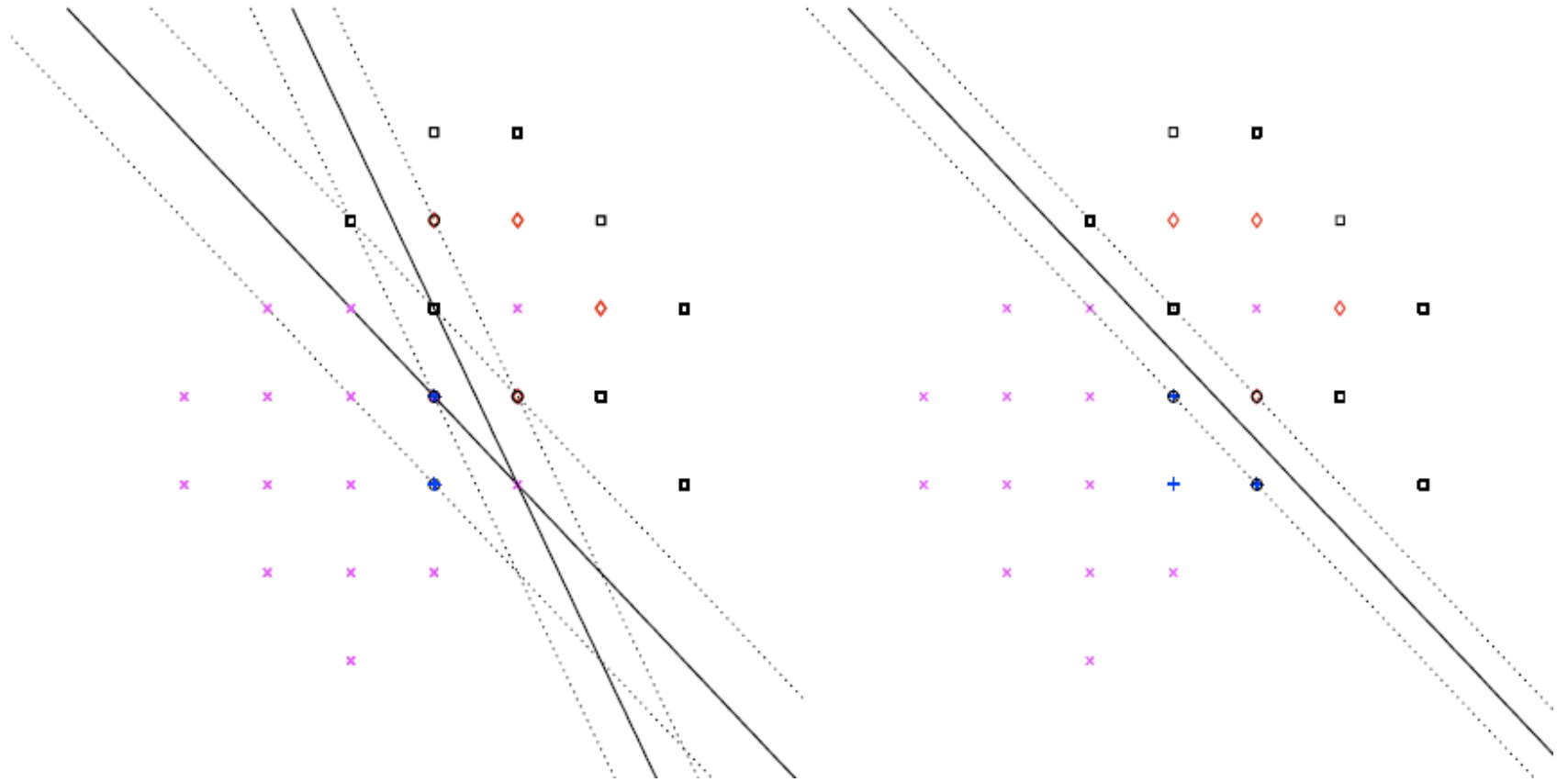
active learning with SVM



- Consider a two class problem
- The SVM tries to find the best separating hyper- plane
- When all the data is labeled it's easy.

◇	Labeled Class 1 data
+	Labeled Class 2 data
□	Unlabeled Class 1 data
X	Unlabeled Class 2 data
○	Support vectors

Uncertainty Sampling






active learning and SVMs

- For each instance that you pick, you halve the hypothesis space.
- In other words you halve the number of possible concepts that fit the data

Uncertainty Sampling



Topic	SVM – Unc	Equivalent Random size
Earn	86.4	34
Acq	77.0	>100
Money	93.8	50
Grain	95.5	13
Crude	95.26	>100

Avg. test set accuracy on Reuters corpus. 2nd column is accuracy with 10 labeled instances using Uncertainty sampling with SVMs.



Maximum Likelihood Parameter Estimation

$$P(X) \sim \theta$$

- For example $\theta = \mu, \sigma$ for a normal distribution.
- Write this as:

$$P(X|\theta)$$

$$\mathcal{D} = x_1 \dots x_n$$

$$p(\mathcal{D}|\theta) = \prod_{i=1}^n p(x_i|\theta)$$

MLE

Log Likelihood: $l(\theta) = \log p(\mathcal{D}|\theta)$

Maximum Likelihood Estimate:

$$\hat{\theta} = \operatorname{argmax}_{\theta} l(\theta)$$

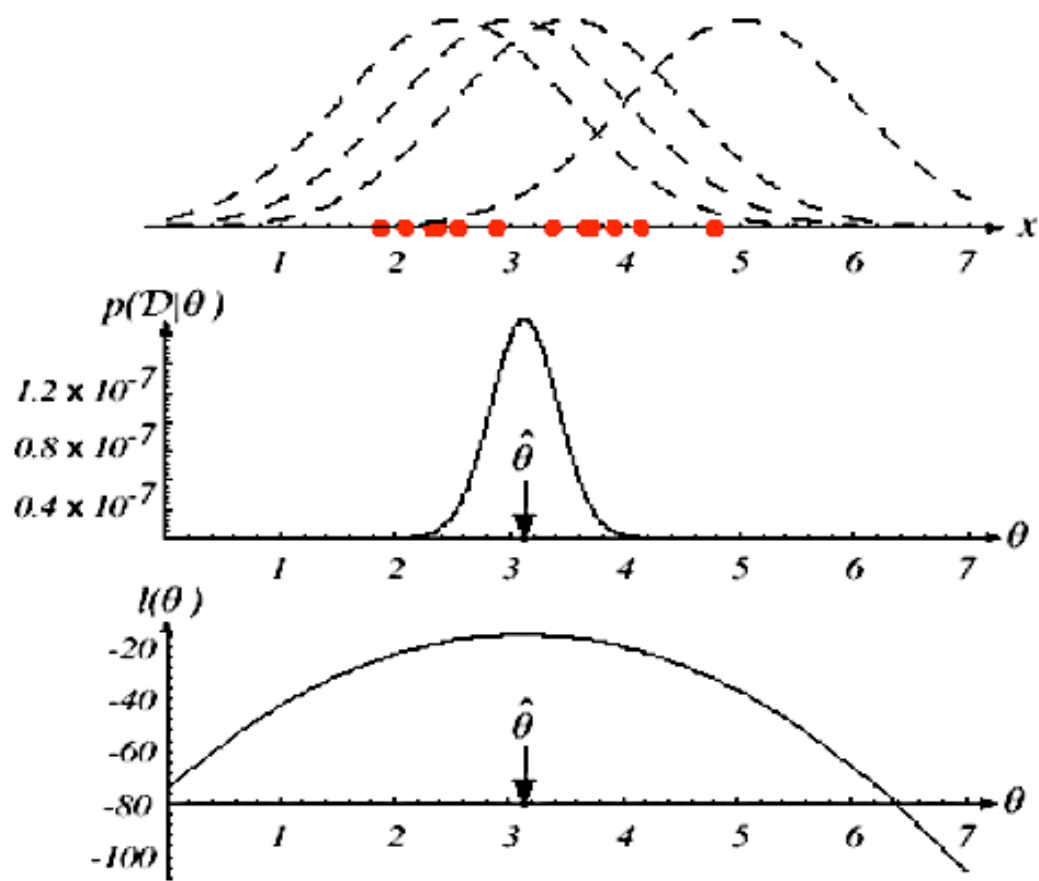


FIGURE 3.1. The top graph shows several training points in one dimension, known or assumed to be drawn from a Gaussian of a particular variance, but unknown mean. Four of the infinite number of candidate source distributions are shown in dashed lines. The middle figure shows the likelihood $p(\mathcal{D}|\theta)$ as a function of the mean. If we had a very large number of training points, this likelihood would be very narrow. The value that maximizes the likelihood is marked $\hat{\theta}$; it also maximizes the logarithm of the likelihood—that is, the log-likelihood $l(\theta)$, shown at the bottom. Note that even though they look similar, the likelihood $p(\mathcal{D}|\theta)$ is shown as a function of θ whereas the conditional density $p(x|\theta)$ is shown as a function of x . Furthermore, as a function of θ , the likelihood $p(\mathcal{D}|\theta)$ is not a probability density function and its area has no significance. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.



Bayesian estimation

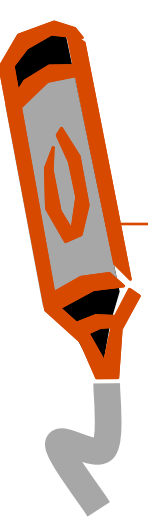
$$P(x|\mathcal{D}) = \int p(x|\theta)p(\theta|\mathcal{D})d\theta$$

$$P(\theta|\mathcal{D}) = \frac{P(\mathcal{D}|\theta)p(\theta)}{\int p(\mathcal{D})}$$

$$P(\theta|\mathcal{D}) = \frac{P(\mathcal{D}|\theta)p(\theta)}{\int p(\mathcal{D}|\theta)p(\theta)d\theta}$$

- used for smoothing language models

text classification



Is this spam?



From: "" jtakworld@hotmail.com
Subject: real estate is the only way... gem oalvgkay

Anyone can buy real estate with no money down

Stop paying rent TODAY !


There is no need to spend hundreds or even thousands for similar courses

I am 22 years old and I have already purchased 6 properties using the methods outlined in this truly INCREDIBLE ebook.

Change your life NOW !

=====
Click Below to order:

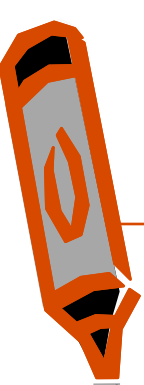
<http://www.wholesaledaily.com/sales/nmd.htm>
=====



Categorization/Classification

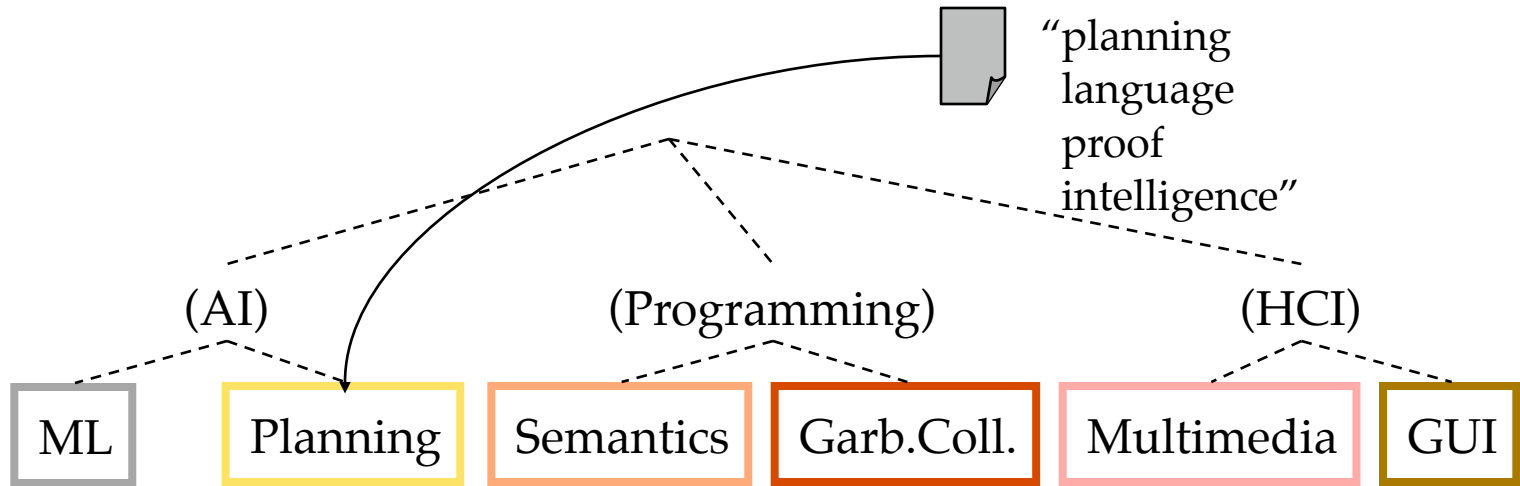
- Given:
 - A description of an instance, $x \in X$, where X is the instance language or instance space.
 - Issue: how to represent text documents.
 - A fixed set of categories:
 $C = \{c_1, c_2, \dots, c_n\}$
- Determine:
 - The category of x : $c(x) \in C$, where $c(x)$ is a categorization function whose domain is X and whose range is C .
 - We want to know how to build categorization functions (“classifiers”).

Document Classification



Testing Data:

Classes:



Training Data:

learning	<u>planning</u>	programming	garbage
<u>intelligence</u>	temporal	semantics	collection		
algorithm	reasoning	<u>language</u>	memory		
reinforcement	plan	<u>proof...</u>	optimization		
network...	<u>language...</u>		region...		

(Note: in real life there is often a hierarchy, not present in the above problem statement; and you get papers on ML approaches to Garb. Coll.)




Text Categorization Examples

Assign labels to each document or web-page:

- Labels are most often topics such as Yahoo-categories
e.g., "finance," "sports," "news;world;asia;business"
- Labels may be genres
e.g., "editorials" "movie-reviews" "news"
- Labels may be opinion
e.g., "like", "hate", "neutral"
- Labels may be domain-specific binary
e.g., "interesting-to-me" : "not-interesting-to-me"
e.g., "spam" : "not-spam"
e.g., "is a toner cartridge ad" : "isn't"

Methods (1)


- 
- Manual classification
 - Used by Yahoo!, Looksmart, about.com, ODP, Medline
 - very accurate when job is done by experts
 - consistent when the problem size and team is small
 - difficult and expensive to scale
 - Automatic document classification
 - Hand-coded rule-based systems
 - Used by CS dept's spam filter, Reuters, CIA, Verity,
 - ...
 - E.g., assign category if document contains a given boolean combination of words
 - Commercial systems have complex query languages (everything in IR query languages + accumulators)



Methods (2)

- Accuracy is often very high if a query has been carefully refined over time by a subject expert
- Building and maintaining these queries is expensive
- Supervised learning of document-label assignment function
 - Many new systems rely on machine learning (Autonomy, Kana, MSN, Verity, ...)
 - k-Nearest Neighbors (simple, powerful)
 - Naive Bayes (simple, common method)
 - Support-vector machines (new, more powerful)
 - ... plus many other methods
 - No free lunch: requires hand-classified training data
 - But can be built (and refined) by non-experts

Text Categorization:

- 
- Representations of text are very high dimensional (one feature for each word).
 - High-bias algorithms that prevent overfitting in high-dimensional space are best.
 - For most text categorization tasks, there are many irrelevant and many relevant features.
 - Methods that combine evidence from many or all features (e.g. naive Bayes, kNN, neural-nets) tend to work better than ones that try to isolate just a few relevant features (standard decision-tree or rule induction)*

*Although one can compensate by using many rules



Bayesian Methods

- Learning and classification methods based on probability theory.
- Bayes theorem plays a critical role in probabilistic learning and classification.
- Build a generative model that approximates how data is produced
- Uses prior probability of each category given no information about an item.
- Categorization produces a posterior probability distribution over the possible categories given a description of an item.



Naive Bayes Classifiers

Task: Classify a new instance based on a tuple of attribute values

$$\langle x_1, x_2, \dots, x_n \rangle$$

$$c_{MAP} = \operatorname{argmax}_{c_j \in C} P(c_j | x_1, x_2, \dots, x_n)$$

$$c_{MAP} = \operatorname{argmax}_{c_j \in C} \frac{P(x_1, x_2, \dots, x_n | c_j) P(c_j)}{P(x_1, x_2, \dots, x_n)}$$

$$c_{MAP} = \operatorname{argmax}_{c_j \in C} P(x_1, x_2, \dots, x_n | c_j) P(c_j)$$

Naïve Bayes Classifier: Assumptions

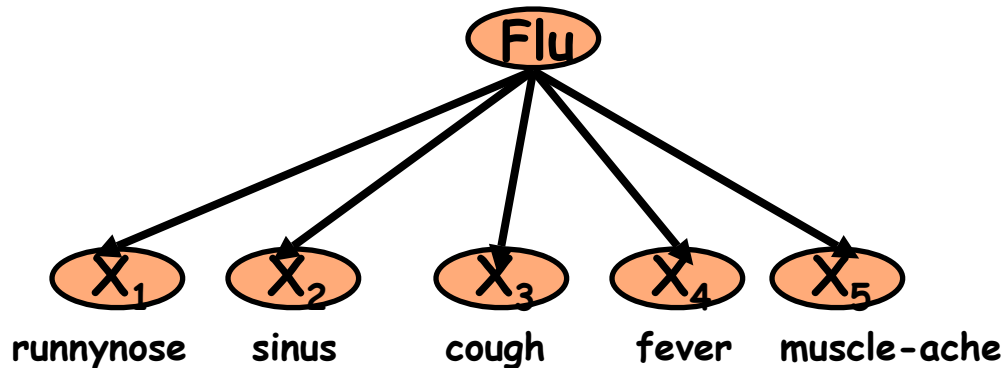


- $P(c_j)$
 - Can be estimated from the frequency of classes in the training examples.
- $P(x_1, x_2, \dots, x_n | c_j)$
 - $O(|X|^n \cdot |C|)$
 - Could only be estimated if a very, very large number of training examples was available.

Conditional Independence Assumption:

⇒ Assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities.

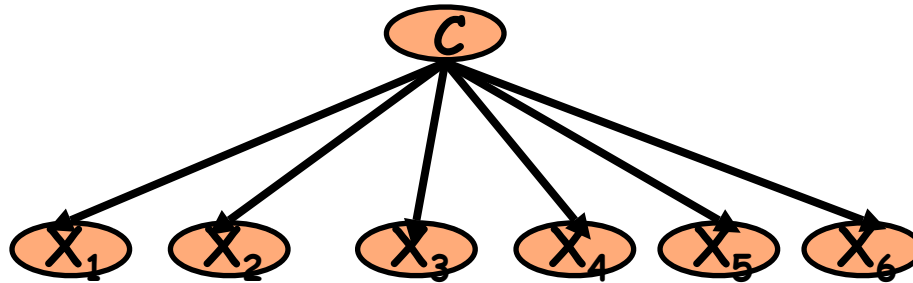
The Naïve Bayes Classifier



- **Conditional Independence Assumption:** features are independent of each other given the class:

$$P(X_1, \dots, X_5 | C) = P(X_1 | C) \cdot P(X_2 | C) \cdot \dots \cdot P(X_5 | C)$$

Learning the Model

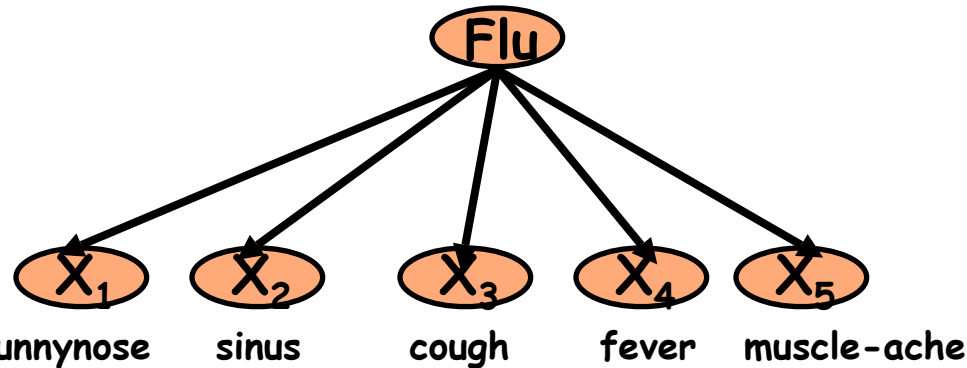


- Common practice: maximum likelihood
 - simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{N(C = c_j)}{N}$$

$$\hat{P}(x_i | c_j) = \frac{N(X_i = x_i, C = c_j)}{N(C = c_j)}$$

Problem with Max Likelihood




$$P(X_1, \dots, X_5 | C) = P(X_1 | C) \cdot P(X_2 | C) \cdot \dots \cdot P(X_5 | C)$$

- What if we have seen no training cases where patient had no flu and muscle aches?

- Zero probabilities cannot be conditioned away, no matter the other evidence!
$$\hat{P}(X_5 = t | C = nf) = \frac{N(X_5 = t, C = nf)}{N(C = nf)} = 0$$

$$\ell = \arg \max_c \hat{P}(c) \prod_i \hat{P}(x_i | c)$$

Smoothing to Avoid Overfitting


$$\hat{P}(x_i | c_j) = \frac{N(X_i = x_i, C = c_j) + 1}{N(C = c_j) + k}$$

of values of X_i

- Somewhat more subtle version

overall fraction in data where $X_i = x_{i,k}$

$$\hat{P}(x_{i,k} | c_j) = \frac{N(X_i = x_{i,k}, C = c_j) + mp_{i,k}}{N(C = c_j) + m}$$

extent of "smoothing"



Naive Bayes Text Classification

- Attributes are text positions, values are words.

$$\begin{aligned}c_{NB} &= \operatorname{argmax}_{c_j \in C} P(c_j) \prod_i P(x_i | c_j) \\ &= \operatorname{argmax}_{c_j \in C} P(c_j) P(x_1 = \text{"our"} | c_j) \cdots P(x_n = \text{"text"} | c_j)\end{aligned}$$



Naive Bayes Text Classification

- Attributes are text positions, values are words.

$$\begin{aligned}c_{NB} &= \operatorname{argmax}_{c_j \in C} P(c_j) \prod_i P(x_i | c_j) \\ &= \operatorname{argmax}_{c_j \in C} P(c_j) P(x_1 = \text{"our"} | c_j) \cdots P(x_n = \text{"text"} | c_j)\end{aligned}$$

- Still too many possibilities



Naive Bayes Text Classification

- Attributes are text positions, values are words.

$$\begin{aligned}c_{NB} &= \operatorname{argmax}_{c_j \in C} P(c_j) \prod_i P(x_i | c_j) \\ &= \operatorname{argmax}_{c_j \in C} P(c_j) P(x_1 = \text{"our"} | c_j) \cdots P(x_n = \text{"text"} | c_j)\end{aligned}$$

- Still too many possibilities
- Assume that classification is independent of the positions of the words




Naive Bayes Text Classification

- Attributes are text positions, values are words.

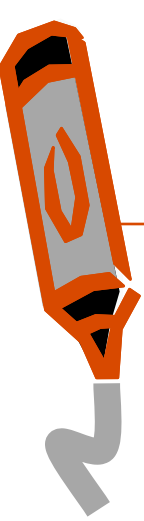
$$\begin{aligned}c_{NB} &= \operatorname{argmax}_{c_j \in C} P(c_j) \prod_i P(x_i | c_j) \\ &= \operatorname{argmax}_{c_j \in C} P(c_j) P(x_1 = \text{"our"} | c_j) \cdots P(x_n = \text{"text"} | c_j)\end{aligned}$$

- Still too many possibilities
- Assume that classification is independent of the positions of the words
 - Use same parameters for each position

Text Classification Algorithms: Learning

- 
- From training corpus, extract *Vocabulary*
 - Calculate required $P(c_j)$ and $P(x_k | c_j)$ terms
 - For each c_j in C do
 - $docs_j \leftarrow$ subset of documents for which the target class is c_j
 - $$P(c_j) = \frac{|docs_j|}{|\text{total \# documents}|}$$
 - $Text_j \leftarrow$ single document containing all $docs_j$
 - for each word x_k in *Vocabulary*
 - $n_k \leftarrow$ number of occurrences of x_k in $Text_j$
 - $$P(x_k | c_j) = \frac{n_k + 1}{n + |Vocabulary|}$$

Text Classification Algorithms: Classifying



- Return c_{NB} , where

$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod P(x_i | c_j)$$



General Learning Issues

- Many hypotheses are usually consistent with the training data.
 - Can derive many classification schemes
- Classification accuracy (% of instances classified correctly).
 - Measured on independent test data.
- Training time (efficiency of training algorithm).
- Testing time (efficiency of subsequent classification).



Text Categorization

Assigning documents to a fixed set of categories.

Applications:


- Web pages
 - Recommending
 - Yahoo-like classification
- Newsgroup Messages
 - Recommending
 - spam filtering
- News articles
 - Personalized newspaper
- Email messages
 - Routing
 - Prioritizing
 - Folderizing
 - spam filtering



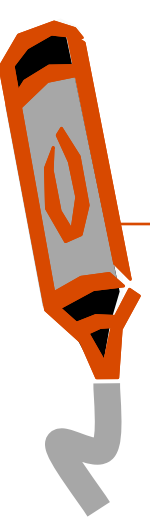
Learning for Text Categorization

- Manual development of text categorization functions is difficult.
- Learning Algorithms:
 - Bayesian (naïve)
 - Neural network
 - Relevance Feedback (Rocchio)
 - Rule based (Ripper)
 - Nearest Neighbor (case based)
 - Support Vector Machines (SVM)

Using Relevance Feedback (Rocchio)

- 
- Relevance feedback methods can be adapted for text categorization.
 - Use standard TF/IDF weighted vectors to represent text documents (normalized by maximum term frequency).
 - For each category, compute a prototype vector by summing the vectors of the training documents in the category.
 - Assign test documents to the category with the closest prototype vector based on cosine similarity.

Rocchio Text Categorization Algorithm(Training)



Assume the set of categories is $\{c_1, c_2, \dots, c_n\}$

For i from 1 to n let $\mathbf{p}_i = \langle 0, 0, \dots, 0 \rangle$ (*init. prototype vectors*)

For each training example $\langle x, c(x) \rangle \in D$

Let \mathbf{d} be the frequency normalized TF/IDF term vector for doc x

Let $i = j: (c_j = c(x))$

(sum all the document vectors in c_i to get \mathbf{p}_i)

Let $\mathbf{p}_i = \mathbf{p}_i + \mathbf{d}$

One vector per category

Rocchio Text Categorization Algorithm (Test)



Given test document x

Let \mathbf{d} be the TF/IDF weighted term vector for x

Let $m = -2$ (*init. maximum cosSim*)

For i from 1 to n :

(compute similarity to prototype vector)

Let $s = \text{cosSim}(\mathbf{d}, \mathbf{p}_i)$

if $s > m$

 let $m = s$

 let $r = c_i$ (*update most similar class prototype*)

Return class r