Text Categorization

Overview

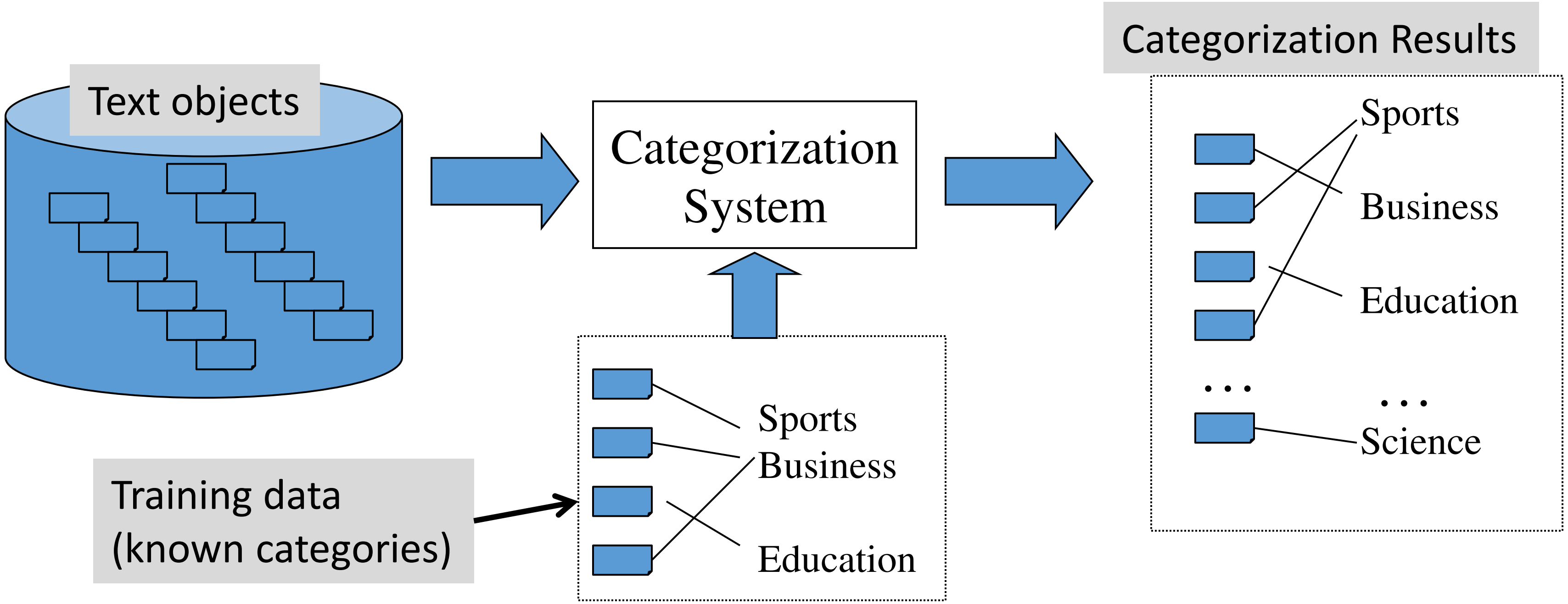
- What is text categorization?
- Why text categorization?
- How to do text categorization?
- How to do feature selection for text categorization?
- How to evaluate categorization results?

Overview

- What is text categorization?
- Why text categorization?
- How to do text categorization?
- How to do feature selection for text categorization?
- How to evaluate categorization results?

Text Categorization

- Given the following:
  - A set of **predefined categories**, possibly forming a hierarchy
  - A **training set** of labeled text objects
- Task: **Classify** a text object into **one or more** of the **categories**



Examples of Text Categorization

- **Text objects can vary** (e.g., documents, passages, or collections of text)
- **Categories can also vary**
  - “**Internal**” categories that characterize a text object (e.g., topical categories, sentiment categories)
  - “**External**” categories that characterize an entity associated with the text object (e.g., author attribution)
- Some **examples of applications**
  - News categorization, literature article categorization (e.g., MeSH annotations)
  - Spam email detection/filtering
  - Sentiment categorization of product reviews or tweets
  - Automatic email sorting/routing
  - Author attribution

Variants of Problem Formulation

- **Binary** categorization: only two categories
  - Retrieval: {relevant-doc, non-relevant-doc}
  - Spam filtering: {spam, non-spam}
  - Opinion: {positive, negative}
- **K-category** categorization: more than two categories
  - Topic categorization: {sports, science, travel, business,…}
  - Email routing:{folder1, folder2,folder3, …}
- **Hierarchical** categorization: Categories form a hierarchy

Binary categorization can potentially support all other categorizations

Overview

- What is text categorization?
- Why text categorization?
- How to do text categorization?
- How to do feature selection for text categorization?
- How to evaluate categorization results?

Why Text Categorization?

- To **enrich text representation** (more understanding of text)
  - Text can now be represented in multiple levels (keywords + categories)
  - Semantic categories assigned can be directly or indirectly useful for an application
  - Semantic categories facilitate aggregation of text content (e.g., aggregating all positive/negative opinions about a product)
- To **infer properties of entities** associated with text data (discovery of **knowledge about the world**)
  - As long as an entity can be associated with text data, we can always use the text data to help categorize the associated entities
  - E.g., discovery of non-native speakers of a language

Overview

- What is text categorization?
- Why text categorization?
- How to do text categorization?
- How to do feature selection for text categorization?
- How to evaluate categorization results?

Categorization Methods: Manual

- Determine the categories based on rules that are carefully designed to reflect the domain knowledge about the categorization problem
- Works well when
  - The categories are very well defined
  - Categories are easily distinguished based on surface features in text (e.g., special vocabulary is known to only occur in a particular category)
  - Sufficient domain knowledge is available to suggest many effective rules
- Problems
  - Labor intensive $\rightarrow$ doesn't scale up well
  - Can't handle uncertainty in rules; rules may be inconsistent $\rightarrow$ not robust
- Both problems can be solved/alleviated by using machine learning

Feature-based Categorization Methods: "Automatic"

- Use **human experts** to
  - Annotate data sets with **category labels** $\rightarrow$ Training data
  - Provide a set of **features** to represent each text object that can potentially provide a "clue" about the category
- Use **machine learning** to learn "soft rules" for categorization from the training data
  - Figure out **which features are most useful** for separating different categories
  - **Optimally combine the features** to **minimize the errors** of categorization on the training data
  - The trained classifier can then be applied to a new text object to predict the most likely category (that a human expert would assign to it)

Machine Learning for Text Categorization

- **General setup:** learn a classifier f: $X \rightarrow Y$
  - Input: X = all text objects; Output: Y = all categories
  - Learn a classifier function, f: $X \rightarrow Y$, such that f(x)=y, $y \in Y$ gives the correct category for $x \in X$ (“correct” is based on the training data)
- **All feature-based methods**
  - Rely on discriminative features of text objects to distinguish categories
  - Combine multiple features in a weighted manner
  - Adjust weights on features to minimize errors on the training data
- **Different methods** tend to vary in
  - Their way of measuring the errors on the training data (may optimize different objective/loss/cost function)
  - Their way of combining features (e.g., linear vs. non-linear)

Generative vs. Discriminative Classifiers

- **Generative** classifiers (learn **what the data “looks” like in each category**)
  - Attempt to model p(X, Y) = p(Y)p(X|Y) and compute p(Y|X) based on p(X|Y) and p(Y) using Bayes Rule
  - Objective function is likelihood, thus indirectly measuring training errors
  - E.g., Naïve Bayes
- **Discriminative** classifiers (learn **what features separate categories**)
  - Attempt to model p(Y|X) directly
  - Objective function directly measures errors of categorization on training data
  - E.g., Logistic Regression, Support Vector Machine (SVM), k-Nearest Neighbor (kNN)

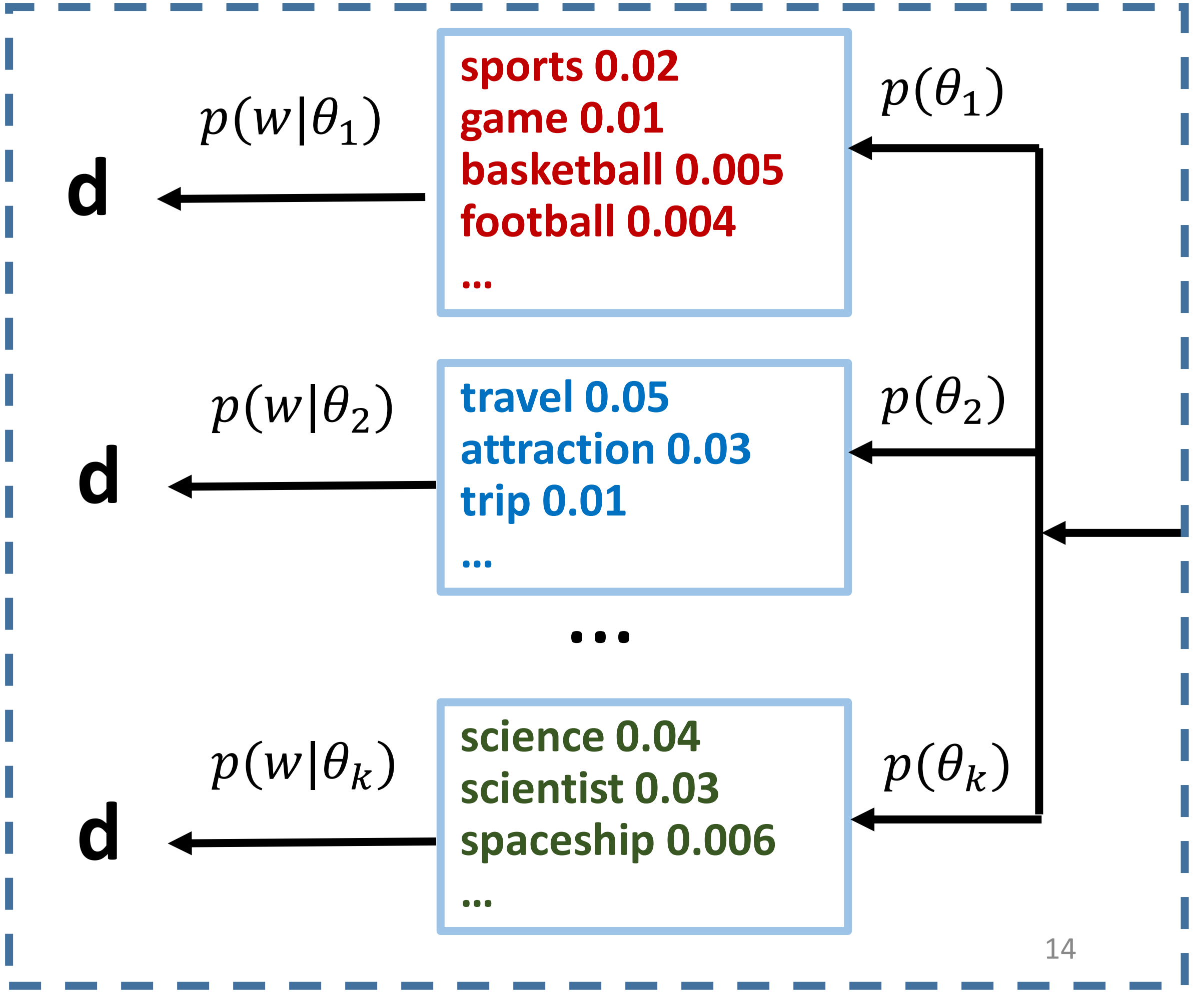
Document Clustering Revisited

Which cluster does d belong to? $\rightarrow$ Which θ_i was used to generate d?

$d = x_1 x_2 \ldots x_L$ where $x_i \in V$

$$cluster(d) = \arg\max_i p(\theta_i|d)$$
$$= \arg\max_i p(d|\theta_i)p(\theta_i)$$
$$= \arg\max_i \prod_{j=1}^{L} p(x_j|\theta_i)p(\theta_i)$$
$$= \arg\max_i \prod_{w \in V} p(w|\theta_i)^{c(w,d)} p(\theta_i)$$

$$p(\theta_i|d) = \frac{p(d|\theta_i)p(\theta_i)}{p(d)} = \frac{p(d|\theta_i)p(\theta_i)}{\sum_{j=1}^{k} p(d|\theta_j)p(\theta_j)}$$



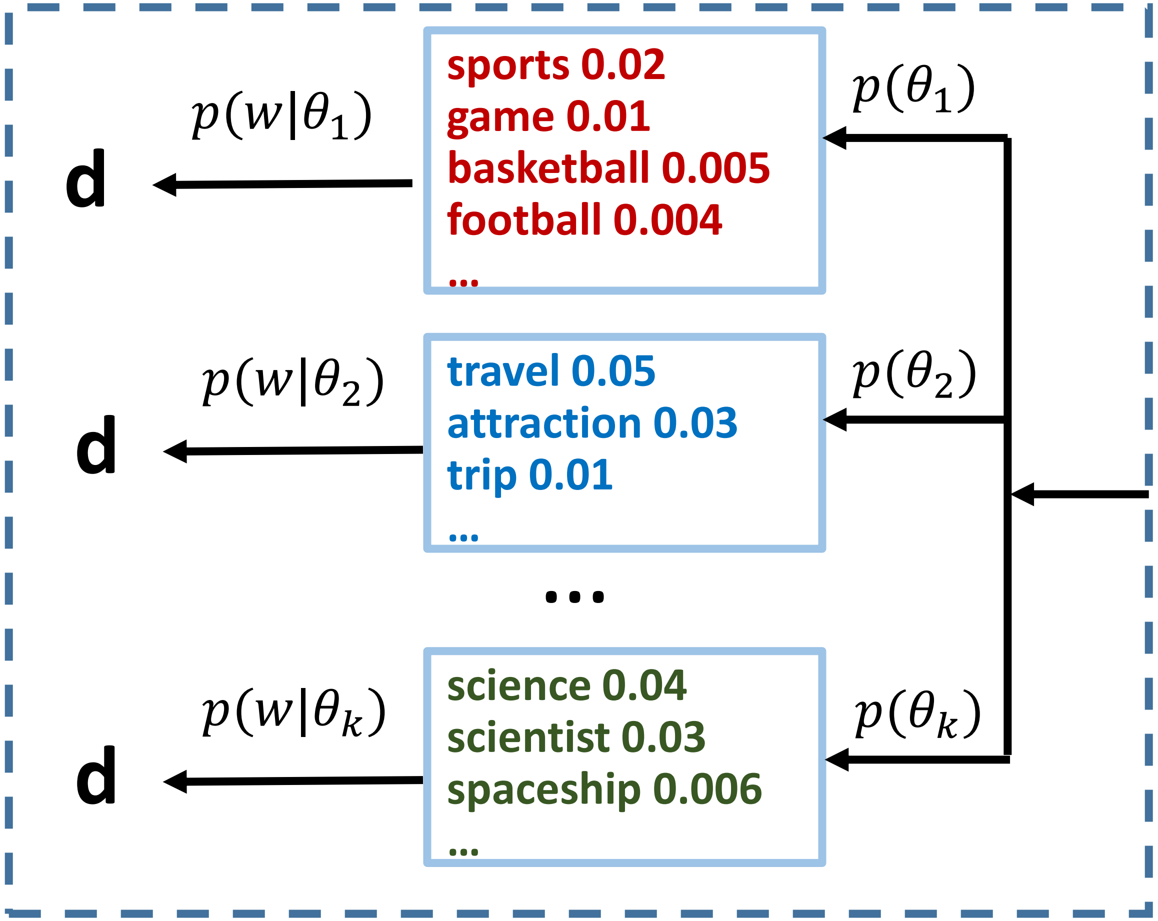
Text Categorization with Naïve Bayes Classifier

$d = x_1 x_2 \dots x_L$ where $x_i \in V$

If $\boldsymbol{\theta_i}$ represents category $\boldsymbol{i}$ accurately, then …

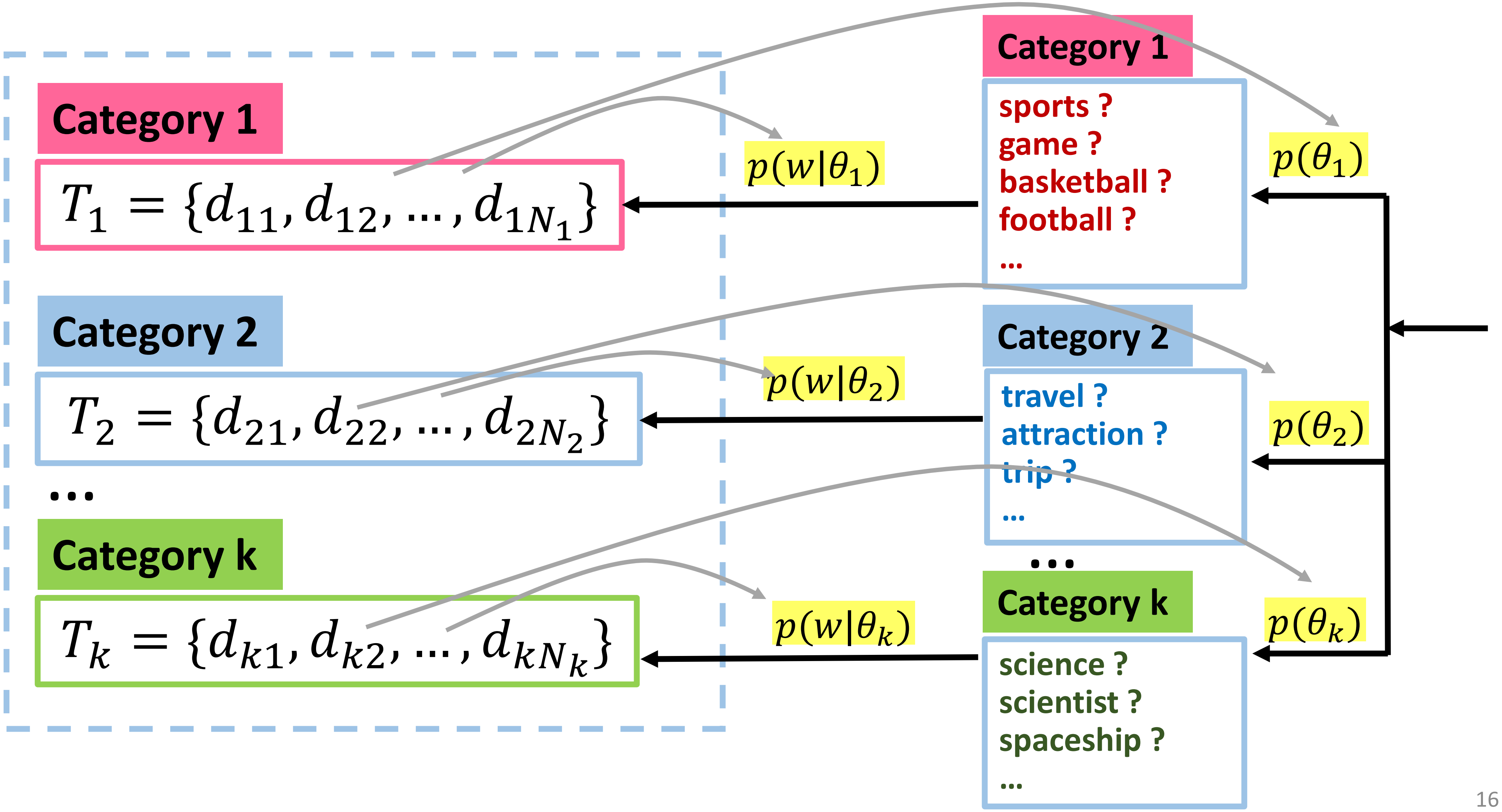
How can we make this happen?

$$category(d) = \arg\max_i p(\theta_i|d)$$
$$= \arg\max_i p(d|\theta_i)p(\theta_i)$$
$$= \arg\max_i \prod_{w \in V} p(w|\theta_i)^{c(w,d)} p(\theta_i)$$



$$category(d) = \arg\max_i \log p(\theta_i) + \sum_{w \in V} c(w,d) \log p(w|\theta_i)$$

Learn from the Training Data



Naïve Bayes Classifier: $p(\theta_i) = ?$ and $p(w|\theta_i) = ?$

Category 1

$$T_1 = \{d_{11}, d_{12}, \dots, d_{1N_1}\}$$

Category 2

$$T_2 = \{d_{21}, d_{22}, \dots, d_{2N_2}\}$$

…

Category k

$$T_k = \{d_{k1}, d_{k2}, \dots, d_{kN_k}\}$$

Which category is most popular?

$$p(\theta_i) = \frac{N_i}{\sum_{j=1}^{k} N_j} \propto |T_i|$$

$$p(w|\theta_i) = \frac{\sum_{j=1}^{N_i} c(w, d_{ij})}{\sum_{w' \in V} \sum_{j=1}^{N_i} c(w', d_{ij})} \propto c(w, T_i)$$

Which word is most frequent in category i?

What are the constraints on $p(\theta_i)$ and $p(w|\theta_i)$?

Smoothing in Naïve Bayes

- Why smoothing?
  - Address data sparseness (training data is small $\rightarrow$ zero probability)
  - Incorporate prior knowledge
  - Achieve discriminative weighting (i.e., IDF weighting)
- How?

$$p(\theta_i) = \frac{N_i + \delta}{\sum_{j=1}^{k} N_j + k\delta} \qquad \delta \geq 0$$

What if $\delta \rightarrow \infty$?

$\boldsymbol{p(w|\theta_B)}$**: background LM**

$$p(w|\theta_i) = \frac{\sum_{j=1}^{N_i} c(w, d_{ij}) + \mu p(w|\theta_B)}{\sum_{w' \in V} \sum_{j=1}^{N_i} c(w', d_{ij}) + \mu} \qquad \mu \geq 0$$

$\boldsymbol{p(w|\theta_B) = 1/|V|}$?

What if $\boldsymbol{\mu} \rightarrow \infty$

Anatomy of Naïve Bayes Classifier

Two categories: $\boldsymbol{\theta_1}$ and $\boldsymbol{\theta_2}$

$$score(d) = \log \frac{p(\theta_1|d)}{p(\theta_2|d)} = \log \frac{p(\theta_1)\prod_{w\in V} p(w|\theta_1)^{c(w,d)}}{p(\theta_2)\prod_{w\in V} p(w|\theta_2)^{c(w,d)}}$$

$$= \boxed{\log \frac{p(\theta_1)}{p(\theta_2)}} + \sum_{w\in V} \underline{c(w,d)} \log \boxed{\frac{p(w|\theta_1)}{p(w|\theta_2)}}$$

Weight on each word (feature) $\boldsymbol{\beta_i}$

Category bias ($\boldsymbol{\beta_0}$) doesn't depend on d!

Sum over all words (features $\{\boldsymbol{f_i}\}$)

Feature value: $\boldsymbol{f_i = c(w,d)}$

Generalize

$$d = (f_1, f_2, .., f_M), \qquad f_i \in \Re$$

$$score(d) = \beta_0 + \sum_{i=1}^{M} f_i \beta_i, \qquad \beta_i \in \Re$$

= Logistic Regression!

Discriminative Classifier 1: Logistic Regression

- Binary Response Variable: $Y \in \{0, 1\}$ Predictors: $X = (x_1, x_2, \dots, x_M), x_i \in \Re$

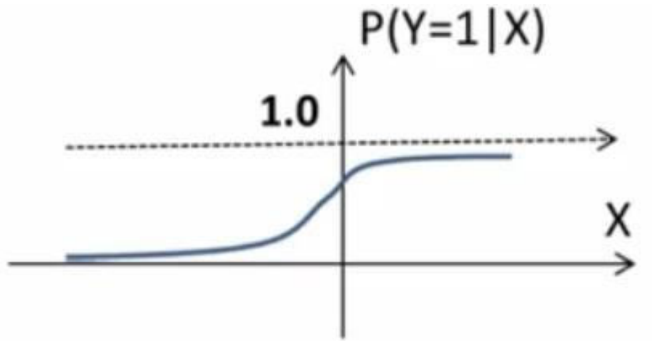
$$Y = \begin{cases} 1 & category(d) = \theta_1 \\ 0 & category(d) = \theta_2 \end{cases}$$

Modeling $p(Y|X)$ directly

Allow many other features than words!

$$\log \frac{p(\theta_1|d)}{p(\theta_2|d)} = \log \frac{p(Y=1|X)}{p(Y=0|X)} = \log \frac{p(Y=1|X)}{1-p(Y=1|X)} = \beta_0 + \sum_{i=1}^{M} x_i \beta_i \qquad \beta_i \in \Re$$

$$p(Y=1|X) = \frac{e^{\beta_0 + \sum_{i=1}^{M} x_i \beta_i}}{e^{\beta_0 + \sum_{i=1}^{M} x_i \beta_i} + 1}$$



Estimation of Parameters

- Training Data: $T = \{(X_i, Y_i)\}, i = 1, 2, \dots, |T|$
- Parameters: $\vec{\beta} = (\beta_0, \beta_1, \dots, \beta_M)$
- Conditional likelihood: $p(T|\vec{\beta}) = \prod_{i=1}^{|T|} p(Y = Y_i | X = X_i, \vec{\beta})$

$Y_i = 1$

$$p(Y = 1|X) = \frac{e^{\beta_0 + \sum_{i=1}^{M} x\beta_i}}{e^{\beta_0 + \sum_{i=1}^{M} x_i\beta_i} + 1}$$

$Y_i = 0$

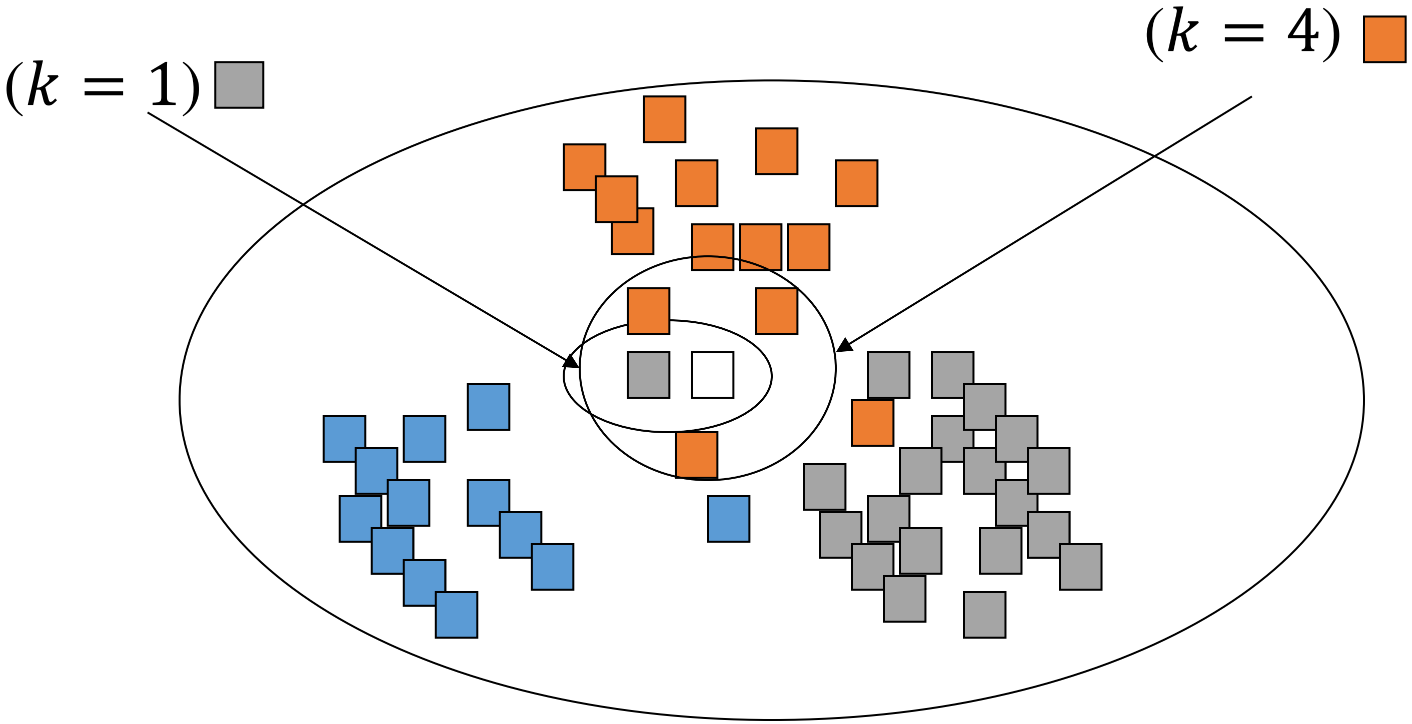
$$p(Y = 0|X) = \frac{1}{e^{\beta_0 + \sum_{i=1}^{M} x_i\beta_i} + 1}$$

- Maximum Likelihood estimate $\vec{\beta}^* = \arg\max_{\vec{\beta}} p(T|\vec{\beta})$
- Can be computed in many ways (e.g., Newton's method)

Discriminative Classifier 2: k-Nearest Neighbors (k-NN)

- Find k examples in the training set that are most similar to the text object to be classified ("neighbor documents")
- Assign the category that is most common in these neighbor text objects (neighbors vote for the category)
- Can be improved by considering the distance of a neighbor (a closer neighbor has more influence)
- Can be regarded as a way to directly estimate the conditional probability of label given data instance, i.e., $p(Y|X)$
- Need a similarity function to measure similarity of two text objects

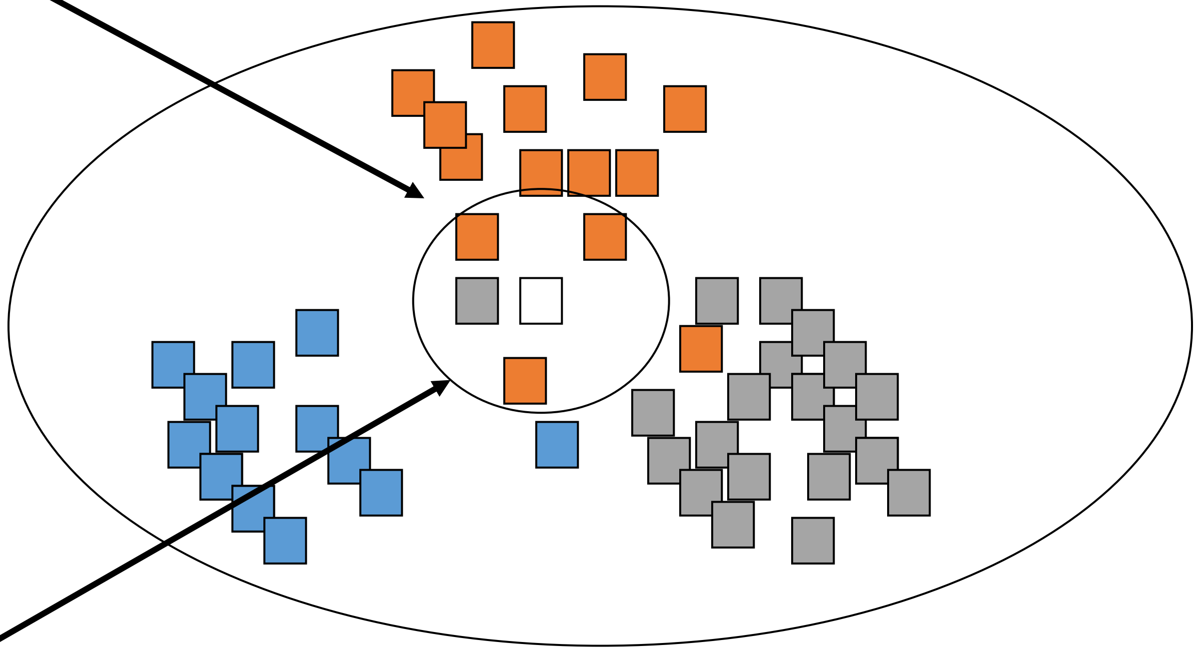
Illustration of K-NN Classifier



k-NN as an Estimate of p(Y|X)

Assume $p(\theta_i|d)$ is locally smooth, i.e., the same for all the d's in this region R

$$p(\theta_i|d) = p(\theta_i|R)$$



Estimate $p(\theta_i|R)$ based on the known categories in the region

$$p(\theta_i|R) = \frac{c(\theta_i, R)}{|R|}$$

Count of d's in R with category θ_i

Total # of docs in R

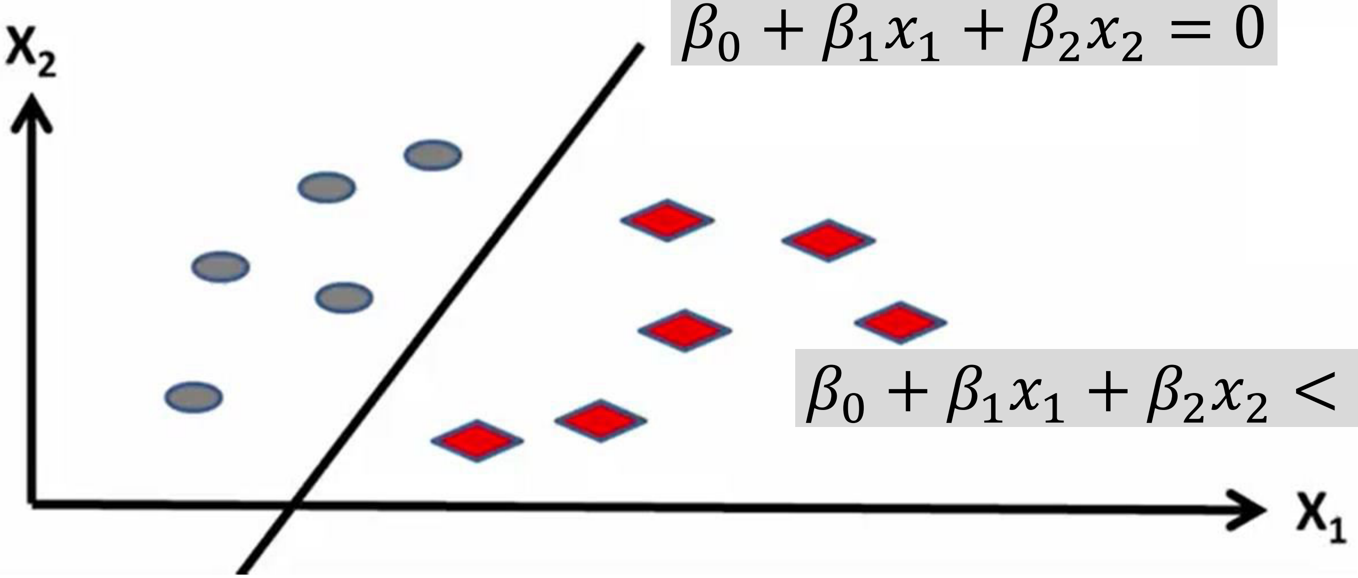
Discriminative Classifier 3: Support Vector Machine (SVM)

$f(X) \geq 0 \Rightarrow X$ is in category θ_1
$f(X) < 0 \Rightarrow X$ is in category θ_2

- Consider two categories: $\{\theta_1, \theta_2\}$
- Use a linear separator $f(X) = \beta_0 + \sum_{i=1}^{M} x_i \beta_i \quad \beta_i \in \Re$

$\beta_0 + \beta_1 x_1 + \beta_2 x_2 > 0$

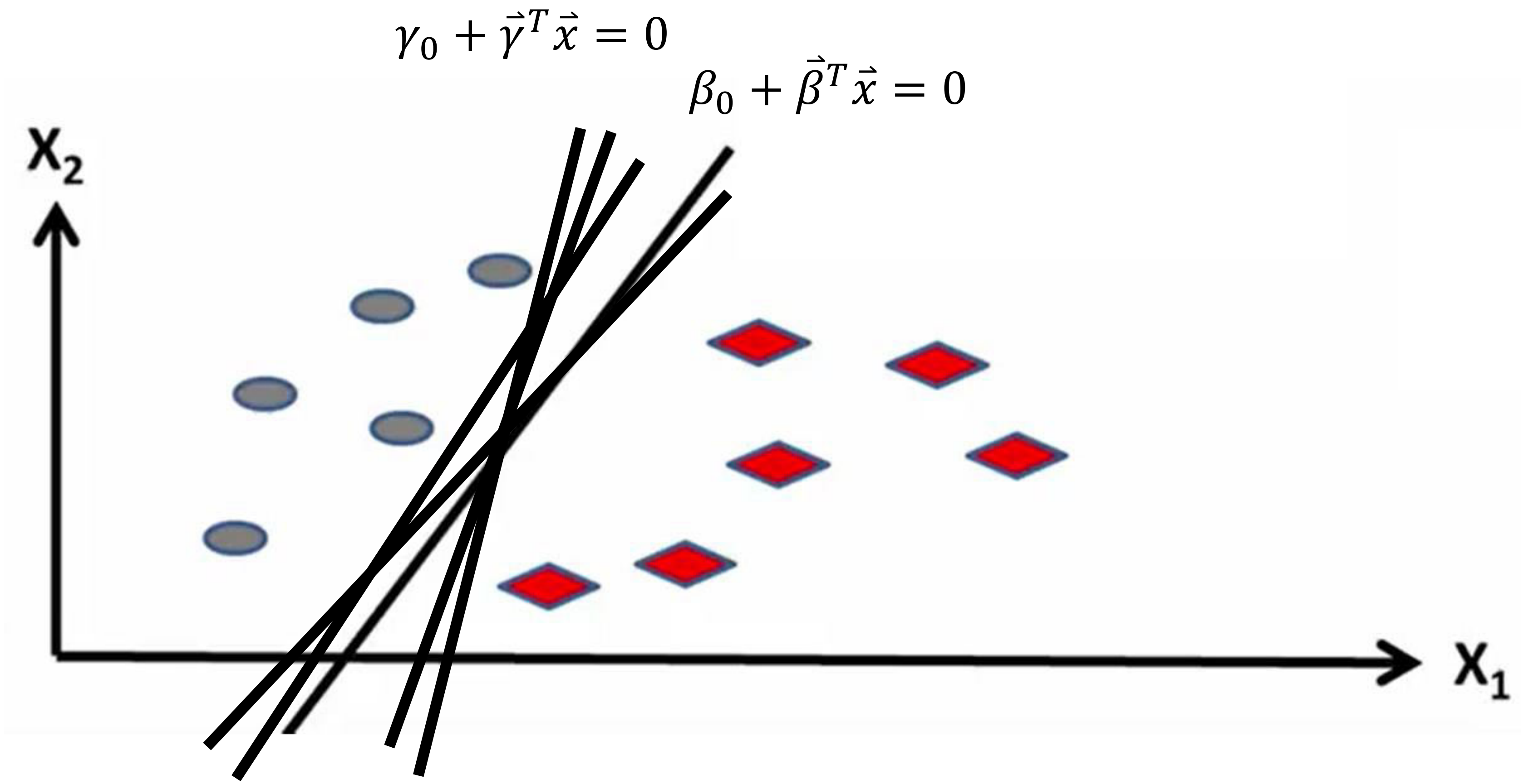
$\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0$



$\beta_0 + \beta_1 x_1 + \beta_2 x_2 < 0$

Assume
$\beta_1 < 0, \beta_2 > 0$

Which Linear Separator Is the Best?

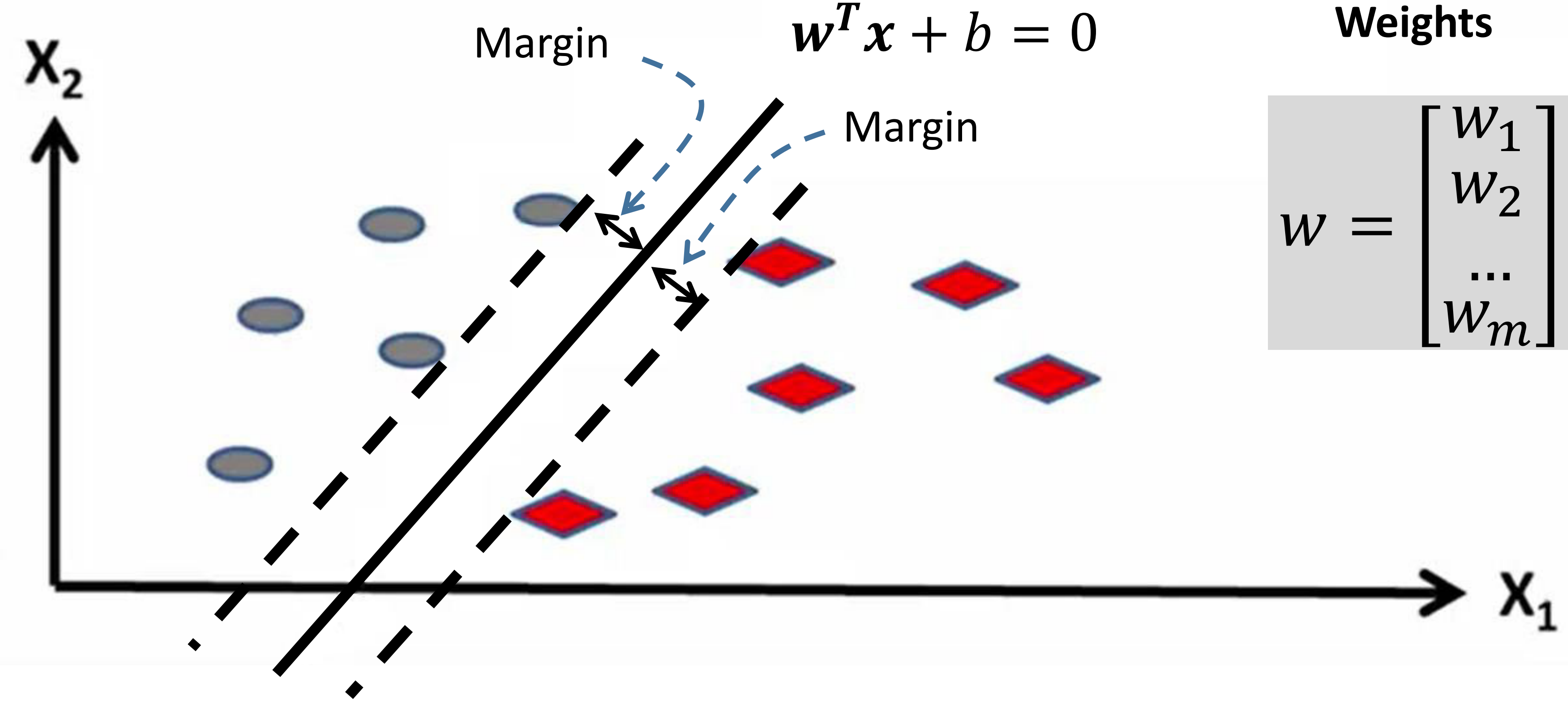


Best Separator = Maximize the Margin

Notation Change: $\boldsymbol{\beta} \rightarrow \boldsymbol{w}; \boldsymbol{\beta_0} \rightarrow \boldsymbol{b}$

Bias Constant

$$\boldsymbol{w}^T\boldsymbol{x} + b = 0$$



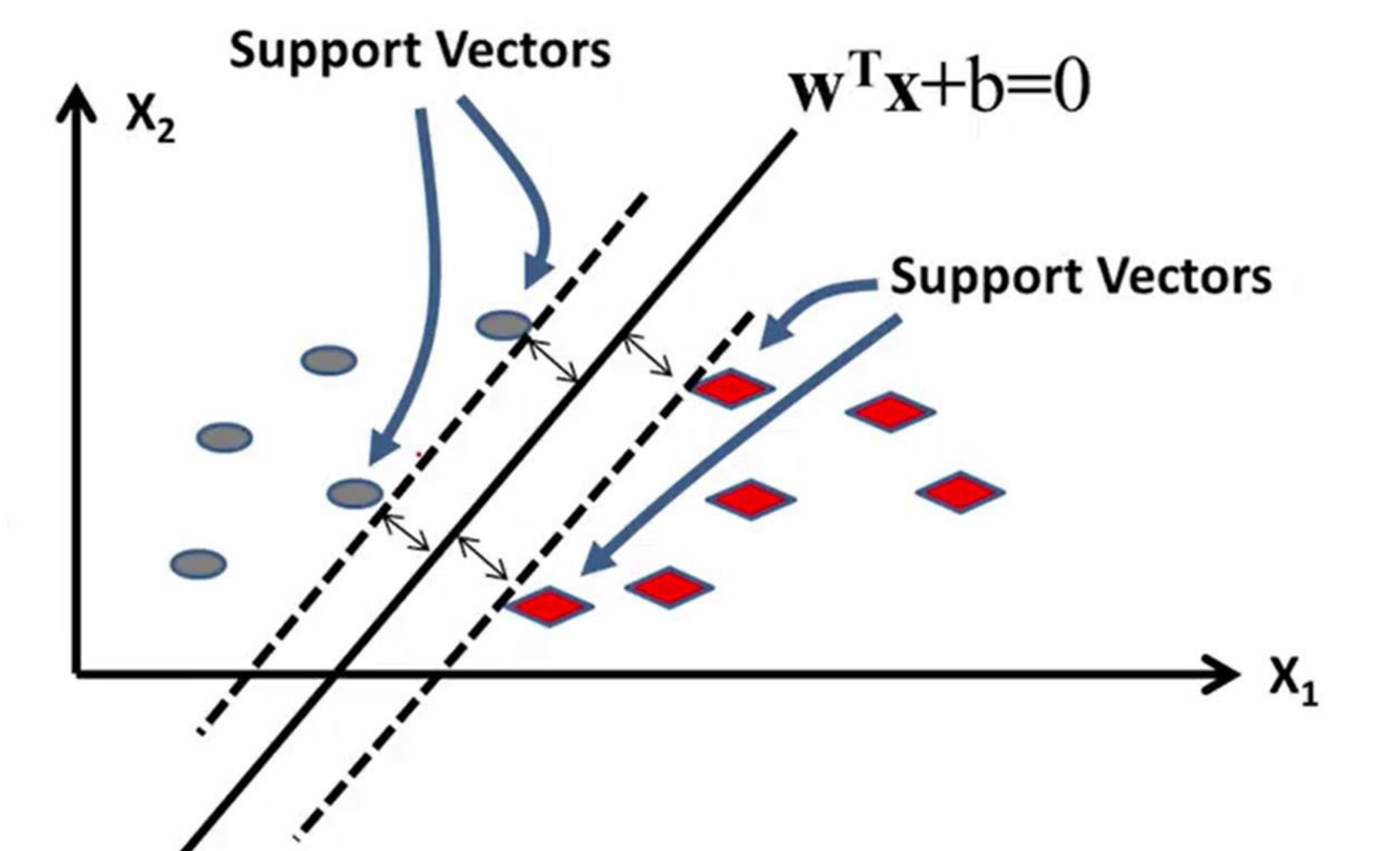
Feature Weights

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_m \end{bmatrix}$$

Feature Vector (e.g., word counts)

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_m \end{bmatrix}$$

Only the Support Vectors Matter



Linear SVM

Classifier: $\mathbf{f}(\mathbf{x}) = \mathbf{w}^{\mathbf{T}}\mathbf{x} + \mathrm{b}$

Parameters: $\mathbf{w}, \mathrm{b}$

Training Data: $\mathrm{T} = \{(\mathrm{x_i}, \mathrm{y_i})\}, \mathrm{i} = 1, \dots, |\mathrm{T}|$

$f(X) \geq 0 \Rightarrow X$ is in category θ_1

$f(X) < 0 \Rightarrow X$ is in category θ_2

$\mathrm{x_i}$ is a feature vector, $\mathrm{y_i} \in \{-1, 1\}$

Goal 1: Correct labeling on training data:

If $\mathbf{y_i} = \mathbf{1} \rightarrow \mathbf{w}^{\mathbf{T}}\mathbf{x_i} + \mathbf{b} \geq \mathbf{1}$

If $\mathbf{y_i} = -\mathbf{1} \rightarrow \mathbf{w}^{\mathbf{T}}\mathbf{x_i} + \mathbf{b} \leq -\mathbf{1}$

Goal 2: Maximize Margin

Large Margin $\Leftrightarrow$ **Small** $\mathbf{w}^{\mathbf{T}}\mathbf{w}$

Constraint

$$\boxed{\forall \mathbf{i}, \mathbf{y_i}\left(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + \mathbf{b}\right) \geq \mathbf{1}}$$

Objective

$$\boxed{\mathbf{Minimize}\ \boldsymbol{\Phi}(\mathbf{w}) = \mathbf{w}^{\mathbf{T}}\mathbf{w}}$$

The optimization problem is quadratic programming with linear constraints

Linear SVM with Soft Margin

Classifier: $\mathbf{f}(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathrm{b} > 0?$

Parameters: $\mathbf{w}, \mathrm{b}$

Added to allow training errors

Training Data: $\mathrm{T} = \{(\mathrm{x_i}, \mathrm{y_i})\}, \mathrm{i} = 1, \dots, |\mathrm{T}|$

Find $\mathbf{w}, \mathbf{b}$**, and** $\boldsymbol{\xi_i}$ **to minimize** $\boldsymbol{\Phi}(\mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{w} \boxed{+ \mathbf{C} \sum_{\boldsymbol{i} \in [\mathbf{1}, |\boldsymbol{T}|]} \boldsymbol{\xi_i}}$

subject to $\forall \boldsymbol{i} \in [\mathbf{1}, |\boldsymbol{T}|], \boldsymbol{y_i}(\boldsymbol{w}^T \boldsymbol{x_i} + \boldsymbol{b}) \geq \mathbf{1} \boxed{- \boldsymbol{\xi_i}, \quad \boldsymbol{\xi_i} \geq \mathbf{0}}$

$C > 0$ is a parameter to control the trade-off between minimizing the errors and maximizing the margin

The optimization problem is still quadratic programming with linear constraints