
Text Categorization using Classical ML





Mausam

(based on slides of Dan Weld, Dan Jurafsky, Prabhakar Raghavan, Hinrich Schutze, Guillaume Obozinski, David D. Lewis, Fei Xia, Michael Collins, Emily Fox, Alexander Ihler, Dan Jurafsky, Dan Klein, Chris Manning, Ray Mooney, Mark Schmidt, Dan Weld, Alex Yates, Luke Zettlemoyer)

Text Classification

- **Given:**
 - A **description of an instance**, $x \in X$, where X is the *instance language* or *instance space*.
 - A **fixed set of categories:**
 $C = \{c_1, c_2, \dots, c_n\}$
 - Notation:
Input (text): x or d
words: w_i
Output (class): y or c
- **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 .

Example: Positive or negative movie sentiment in a review?

-  • unbelievably disappointing
-  • Full of zany characters and richly applied satire, and some great plot twists
-  • this is the greatest screwball comedy ever filmed
-  • It was pathetic. The worst part about it was the boxing scenes.

Example: Academic Literature Categorization

MeSH Subject Category Hierarchy

MEDLINE Article



- Antagonists and Inhibitors

- Blood Supply

- Chemistry

- Drug Therapy

- Embryology

- Epidemiology

- ...

Example: Male or female author?

- The main aim of this article is to propose an exercise in stylistic analysis which can be employed in the teaching of English language. It details the design and results of a workshop activity on narrative carried out with undergraduates in a university department of English. The methods proposed are intended to enable students to obtain insights into aspects of cohesion and narrative structure: insights, it is suggested, which are not as readily obtainable through more traditional methods.
Female writers use
more first person/second person pronouns
more gender laden third person pronouns
(overall more personalization)
- My aim in this article is to provide an approach to understanding of what some of these so-called apposition markers indicate. It will be argued that the decision to put something in other words is essentially a decision about style, a point which is, perhaps, anticipated by Burton-Roberts when he describes loose apposition as a rhetorical device. However, he does not justify this suggestion by giving the criteria for classifying a mode of expression as a rhetorical device.

Classification Method (Gen 1): Hand-coded rules

- Rules based on combinations of words or other features
 - spam: black-list-address OR (“dollars” AND “have been selected”)
- Accuracy can be high
 - If rules carefully refined by expert
- But building and maintaining these rules is expensive
- MACHINE LEARNING

This Lecture: Setting the Basics

- Naïve Bayes
- Logistic Regression

ML Topics I assume you know

- (k fold) Cross-validation
- Accuracy, Precision, Recall, F-score
- Area under precision-recall curve
- Train, dev, test sets

Is this relevant today? (Paper from 2025)

A thorough benchmark of automatic text classification
From traditional approaches to large language models

		Traditional Approaches			Small Language Models (SLMs)				Large Language Models (LLMs)				
dataset		LSVM	LR	RF	RoBERTa	BERT	BART	XLNet	BloomZ	LLaMa 2	LLaMa 3.1	Mistral	DeepSeek
Topic	DBLP	79.7(0.7)	79.2(0.7)	62.6(0.9)	81.4(0.5)	81.7(0.5)	81.1(0.5)	81.4(0.6)	85.9(0.6)	86.8(0.4)	87.8(0.4)	86.7(0.5)	86.5(0.5)
	Books	84.5(0.5)	84.1(0.5)	75.7(0.5)	87.2(0.6)	89.5(0.2)	86.9(0.5)	87.3(0.4)	89.4(0.6)	93.0(0.5)	93.0(0.3)	92.6(0.4)	92.2(0.5)
	ACM	67.7(1.5)	66.7(1.3)	60.1(1.2)	70.3(1.4)	71.8(1.0)	70.8(0.7)	69.9(0.9)	72.9(1.7)	74.9(1.9)	77.8(0.9)	76.3(1.4)	75.2(1.3)
	20NG	89.8(0.6)	89.4(0.7)	81.6(0.5)	86.8(0.7)	85.4(0.5)	87.4(0.9)	87.4(0.8)	87.9(0.6)	89.9(0.6)	90.4(0.6)	90.3(0.7)	89.2(0.7)
	OHSUMED	72.9(1.5)	72.0(1.4)	56.7(1.2)	77.8(1.2)	76.4(1.2)	77.6(0.7)	77.6(1.0)	81.5(1.0)	82.2(0.9)	83.1(1.1)	83.1(0.8)	82.0(0.9)
	Reuters90	33.2(2.3)	41.5(2.6)	27.0(1.6)	41.9(2.2)	40.2(2.8)	42.2(2.1)	41.3(2.6)	40.7(2.2)	41.8(2.5)	41.5(2.6)	41.5(2.4)	41.7(2.7)
	WOS-11967	85.9(0.5)	85.4(0.5)	83.8(0.6)	86.8(0.4)	85.5(0.7)	86.9(0.8)	87.0(0.7)	89.4(0.6)	88.6(0.5)	89.9(0.4)	89.1(0.7)	89.6(0.7)
	WebKB	70.7(1.9)	71.4(2.7)	64.9(1.6)	83.0(2.0)	83.2(2.1)	83.0(1.7)	81.9(2.5)	84.7(1.8)	85.7(1.2)	87.3(1.5)	86.0(1.3)	85.7(1.8)
	Twitter	64.0(1.1)	63.1(1.1)	45.5(1.3)	78.4(1.8)	64.5(1.9)	79.0(2.1)	76.4(2.1)	80.0(1.8)	78.8(1.9)	78.6(1.6)	79.3(2.4)	78.4(1.8)
	TREC	68.8(2.5)	68.7(2.3)	66.0(1.8)	95.5(0.5)	87.6(1.4)	95.5(0.8)	94.3(1.1)	96.0(0.8)	95.9(0.6)	96.1(0.8)	96.0(0.8)	96.1(0.6)
WOS-5736	91.2(0.8)	91.7(0.8)	91.0(0.6)	90.5(0.9)	89.7(1.3)	89.6(1.7)	90.2(0.9)	91.3(0.8)	89.8(0.8)	91.9(0.5)	91.9(0.6)	91.4(0.9)	
Sentiment	SST1	34.8(1.1)	28.9(0.7)	33.6(1.1)	53.8(1.3)	51.6(1.2)	52.8(1.0)	51.4(1.7)	55.8(1.2)	59.3(0.7)	58.7(1.0)	58.1(0.9)	57.6(0.9)
	pang_movie	77.3(0.8)	77.0(1.0)	75.5(0.8)	89.0(0.4)	87.4(0.4)	88.1(0.5)	88.2(0.6)	93.3(0.3)	93.6(0.4)	93.6(0.4)	93.6(0.4)	92.9(0.2)
	Movie Review	75.9(0.9)	75.7(1.2)	73.5(0.5)	89.0(0.7)	87.7(0.5)	88.2(0.6)	86.4(3.3)	96.5(0.4)	93.6(0.3)	92.0(4.0)	93.8(0.5)	92.8(0.4)
	vader_movie	78.6(0.8)	78.5(0.6)	76.4(0.9)	91.3(0.5)	88.2(0.7)	90.4(0.6)	90.5(0.4)	96.2(0.3)	95.7(0.3)	95.8(0.4)	95.9(0.3)	95.1(0.5)
	MPQA	78.8(0.8)	78.5(0.8)	78.3(0.9)	90.2(0.8)	89.1(0.7)	90.1(0.7)	88.6(0.5)	90.6(0.6)	91.5(0.3)	91.5(0.5)	91.4(0.4)	91.1(0.4)
	Subj	89.1(0.6)	88.9(0.5)	87.8(0.7)	96.9(0.4)	97.0(0.3)	96.8(0.4)	96.1(0.5)	97.8(0.3)	98.4(0.3)	98.4(0.3)	98.4(0.3)	98.1(0.4)
	SST2	78.9(0.6)	79.1(0.7)	76.9(0.4)	93.2(0.6)	91.5(0.6)	92.8(0.5)	92.1(0.4)	95.9(0.3)	96.5(0.4)	96.5(0.4)	96.4(0.5)	96.3(0.5)
	yelp_reviews	94.8(0.7)	94.9(0.8)	87.9(0.9)	97.9(0.4)	95.6(0.6)	97.5(0.4)	97.3(0.4)	99.1(0.2)	99.2(0.2)	99.4(0.1)	99.1(0.3)	99.1(0.2)
Large	AGNews	91.8(0.3)	91.5(0.2)	88.1(0.3)	94.2(0.2)	93.9(0.2)	93.9(0.2)	94.0(0.1)	95.7(0.1)	95.4(0.1)	95.7(0.2)	95.0(0.1)	95.6(0.3)
	Yelp_2013	56.8(0.1)	59.1(0.1)	51.0(0.1)	64.4(0.6)	63.6(0.4)	63.8(0.5)	63.0(0.5)	67.7(0.2)	69.3(0.3)	69.5(0.1)	61.3(0.3)	69.1(0.3)
	MEDLINE	80.4(0.4)	79.9(0.6)	80.0(0.6)	81.8(0.6)	75.8(0.8)	82.2(0.2)	60.3(0.5)	85.0(0.2)	85.4(0.2)	85.6(0.3)	84.9(0.2)	85.3(0.2)

Table 2: Effectiveness: Macro-F1 and Confidence Interval (CI) of each classifier. Bold values mean statistical ties (t-test with Bonferroni correction), and green backgrounds highlight the best (i.e., higher absolute value) performance per dataset.

Training + Test time in Seconds

		Traditional Approaches			Small Language Models (SLMs)				Large Language Models (LLMs)				
dataset		LSVM	LR	RF	RoBERTa	BERT	BART	XLNet	BloomZ	LLaMa 2	LLaMa 3.1	Mistral	DeepSeek
Topic	DBLP	118.7	137.5	1017.0	2354.9	1988.0	2693.7	3874.7	17960.7	19054.5	16893.5	17002.4	16993.9
	Books	86.6	85.2	1174.0	2075.1	1790.7	1440.7	3150.0	15847.7	16765.2	14832.2	14822.0	14883.9
	ACM	81.0	194.9	713.5	1430.5	1243.5	1683.7	2385.9	11712.3	12396.4	10964.2	10992.6	11013.3
	20NG	164.2	159.1	1160.9	1305.9	1858.7	1597.6	2034.9	8886.7	9400.5	8349.6	8372.5	8380.4
	OHSUMED	134.4	126.0	743.9	1306.0	1142.5	1427.9	2040.9	8644.9	9160.9	8111.9	8123.5	8144.0
Sentiment	SST1	14.4	10.2	105.4	386.7	581.4	438.5	1585.4	5568.6	5886.3	5217.3	5230.9	5246.9
	pang_movie	15.1	3.5	76.4	320.5	512.7	394.0	560.3	5011.8	5298.7	4693.5	4706.5	4717.9
	Movie Review	10.1	1.7	59.7	316.5	520.9	399.9	1352.2	5007.9	5297.2	4694.6	4706.8	4718.0
	vader_movie	12.4	3.4	73.9	319.0	510.3	376.0	555.7	4966.3	5250.7	4656.7	4665.3	4677.9
	MPQA	0.6	2.2	36.3	318.0	509.3	401.5	1072.5	4985.7	5268.6	4670.0	4673.1	4683.7
Large	AGNews	162.7	406.7	3626.3	8464.0	5468.1	7347.0	10243.6	57728.4	57302.2	51153.5	50981.5	51184.6
	Yelp_2013	6666.6	8319.5	18945.7	18023.5	14967.5	22872.9	31684.8	152396.2	152396.2	135039.3	134468.5	130036.1
	MEDLINE	2610.2	10015.9	26163.8	57354.2	49114.3	90709.8	99641.6	368407.6	389803.8	345407.9	344450.1	345134.4

Table 3: Total application cost (training + test time in seconds) of the classifiers' application in each dataset. Due to length constraints, we only show a selection of datasets; conclusions for others remain similar. All results are accessible on GitHub.

A thorough benchmark of automatic text classification From traditional approaches to large language models

Washington Cunha
Federal University of Minas Gerais
Brazil
washingtoncunha@dcc.ufmg.br

Leonardo Rocha
Federal University of São João del Rei
Brazil
lrocha@ufsj.edu.br

Marcos André Gonçalves
Federal University of Minas Gerais
Brazil
mgoncalv@dcc.ufmg.br

CO₂ generated during training

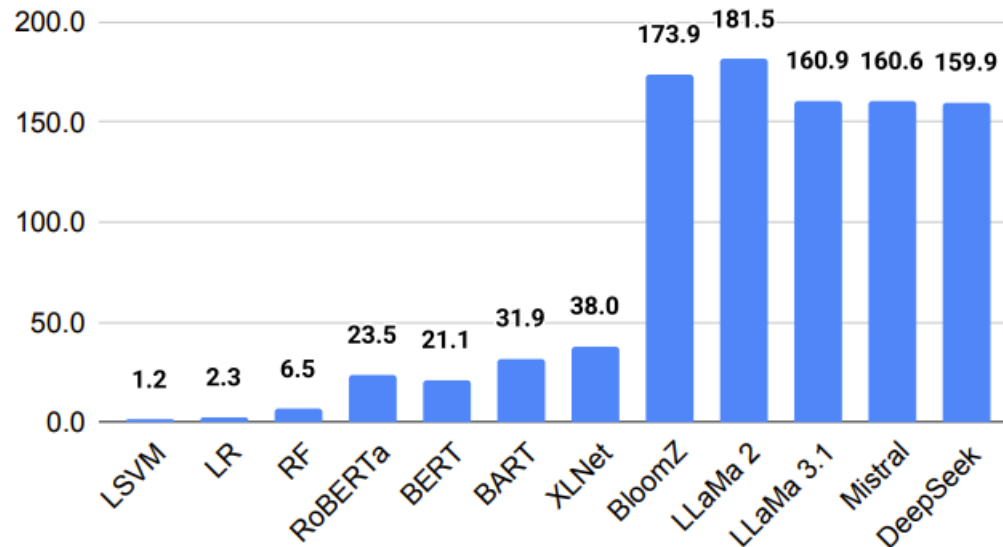


Figure 1: CO₂e: the equivalent amount of carbon dioxide (in kg) generated by the classification models' fine-tuning.

**A thorough benchmark of automatic text classification
From traditional approaches to large language models**

Washington Cunha
Federal University of Minas Gerais
Brazil
washingtoncunha@dcc.ufmg.br

Leonardo Rocha
Federal University of São João del Rei
Brazil
lrocha@ufsj.edu.br

Marcos André Gonçalves
Federal University of Minas Gerais
Brazil
mgoncalv@dcc.ufmg.br

The bag of words representation

Y (

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.

) = C



The bag of words representation

Y(

I **love** this movie! It's **sweet**, but with **satirical** humor. The dialogue is **great** and the adventure scenes are **fun**... It manages to be **whimsical** and **romantic** while **laughing** at the conventions of the fairy tale genre. I would **recommend** it to just about anyone. I've seen it **several** times, and I'm always **happy** to see it **again** whenever I have a friend who hasn't seen it yet.

)=C



The bag of words representation: using a subset of words

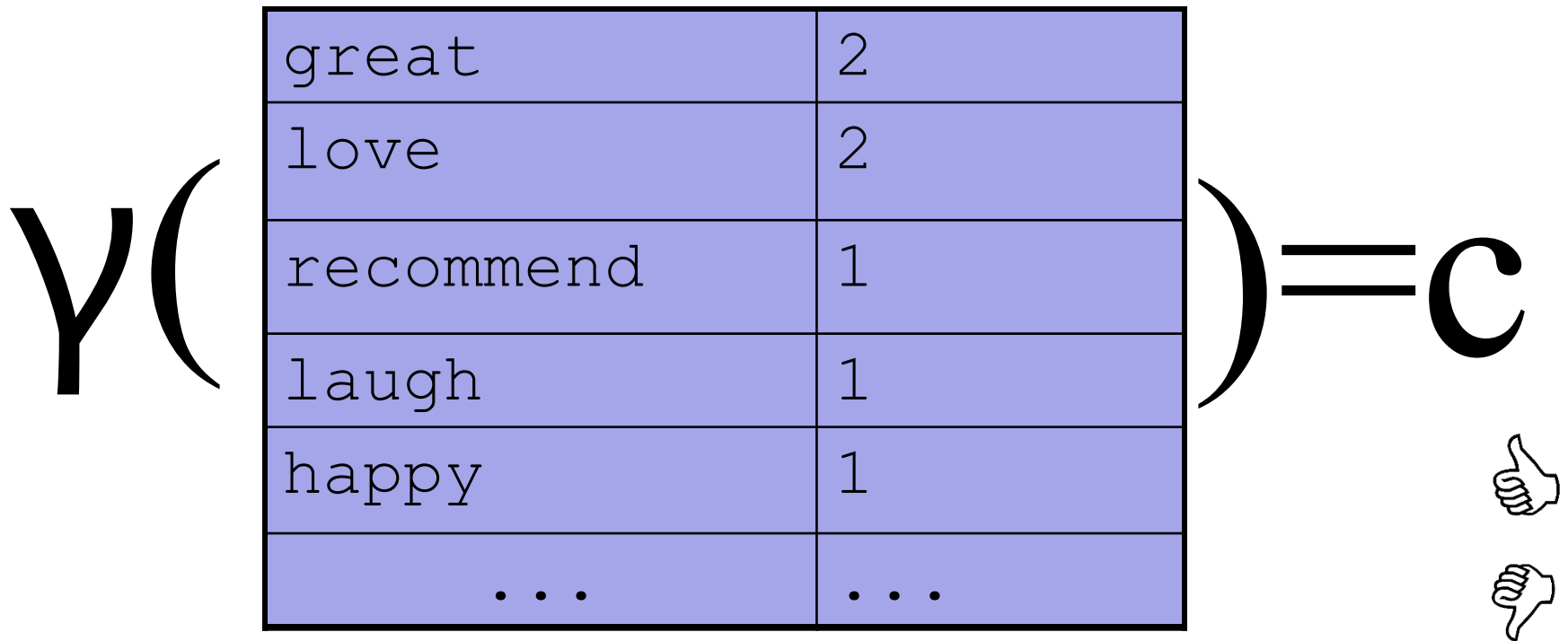
Y (

```
x love xxxxxxxxxxxxxxxxxxxx sweet
xxxxxxxx satirical xxxxxxxxxxxx
xxxxxxxxxxxx great xxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxx fun xxxx
xxxxxxxxxxxxxxxx whimsical xxxx
romantic xxxx laughing
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx recommend xxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xx several xxxxxxxxxxxxxxxxxxxxxxxx
xxxxx happy xxxxxxxxxxxx again
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

) = C



The bag of words representation



Bayes' Rule Applied to Documents and Classes

- For a document d and a class c

$$P(c | d) = \frac{P(d | c)P(c)}{P(d)}$$

Naïve Bayes Classifier (I)

$$c_{MAP} = \operatorname{argmax}_{c \in C} P(c | d)$$

MAP is “maximum a posteriori” = most likely class

$$= \operatorname{argmax}_{c \in C} \frac{P(d | c)P(c)}{P(d)}$$

Bayes Rule

$$= \operatorname{argmax}_{c \in C} P(d | c)P(c)$$

Dropping the denominator

Naïve Bayes Classifier (II)

$$c_{MAP} = \operatorname{argmax}_{c \in C} P(d | c)P(c)$$

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

Document d
represented as
features $x_1 \dots x_n$

Naïve Bayes Classifier (IV)

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

$O(|X|^n \cdot |C|)$ parameters

Could only be estimated if a very, very large number of training examples was available.

How often does this class occur?

We can just count the relative frequencies in a corpus

Multinomial Naïve Bayes Classifier

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

$$c_{NB} = \operatorname{argmax}_{c \in C} P(c) \prod_{x \in X} P(x | c)$$

Multinomial Naïve Bayes Independence Assumptions

$$P(x_1, x_2, \dots, x_n | c)$$

- **Bag of Words assumption:** Assume position doesn't matter
- **Conditional Independence:** Assume the feature probabilities $P(x_i | c_j)$ are independent given the class c .

$$P(x_1, \dots, x_n | c) = P(x_1 | c) \cdot P(x_2 | c) \cdot P(x_3 | c) \cdot \dots \cdot P(x_n | c)$$

Learning the Multinomial Naïve Bayes Model

- Features: presence/absence of word
 - Or each position as random var and word as value
- First attempt: maximum likelihood estimates
 - simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{\text{doccount}(C = c_j)}{N_{doc}}$$

$$\hat{P}(w_i | c_j) = \frac{\text{count}(w_i, c_j)}{\sum_{w \in V} \text{count}(w, c_j)}$$

Problem with Maximum Likelihood

lec10.3

- What if we have seen no training documents with the word ***fantastic*** and classified in the topic **positive (*thumbs-up*)**?

$$\hat{P}(\text{"fantastic"} \mid \text{positive}) = \frac{\text{count}(\text{"fantastic"}, \text{positive})}{\sum_{w \in V} \text{count}(w, \text{positive})} = 0$$

- Zero probabilities cannot be conditioned away, no matter the other evidence!

$$c_{MAP} = \operatorname{argmax}_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$

Laplace (add-1) smoothing for Naïve Bayes

$$\hat{P}(w_i | c) = \frac{\text{count}(w_i, c) + 1}{\sum_{w \in V} (\text{count}(w, c) + 1)}$$
$$= \frac{\text{count}(w_i, c) + 1}{\sum_{w \in V} \text{count}(w, c) + |V|}$$

Easy to Implement

- But...
- If you do... it probably won't work...

Probabilities: Important Detail!

- We are multiplying lots of small numbers
Danger of underflow!
 - $0.5^{57} = 7 \text{ E } -18$
- Solution? Use logs and add!
 - $p_1 * p_2 = e^{\log(p1)+\log(p2)}$
 - Always keep in log form

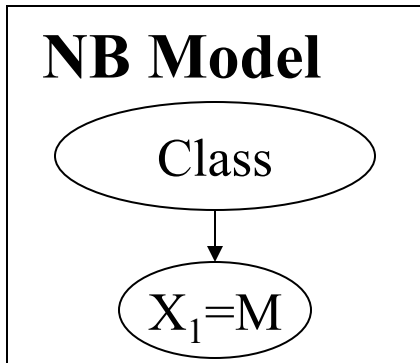
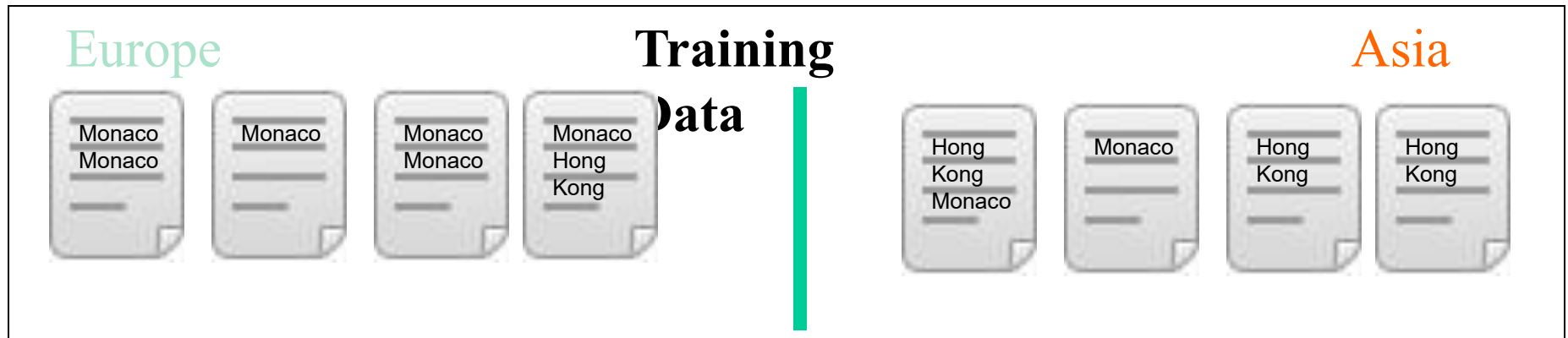
Advantages

- Simple to implement
 - No numerical optimization, matrix algebra, etc
- Efficient to train and use
 - Easy to update with new data
 - Fast to apply
- Binary/multi-class
- Comparatively good effectiveness with small training sets
- A good dependable baseline for text classification
 - But we will see other classifiers that give better accuracy

Disadvantages

- Independence assumption wrong
 - Absurd estimates of class probabilities
 - Output probabilities close to 0 or 1
 - Thresholds must be tuned; not set analytically
- NB is a Generative model
 - We model $P(c, d)$ – can be used to generate docs
 - Generally lower effectiveness than discriminative techniques

Text classification: Asia or Europe



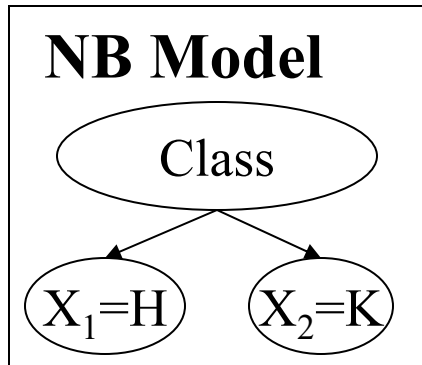
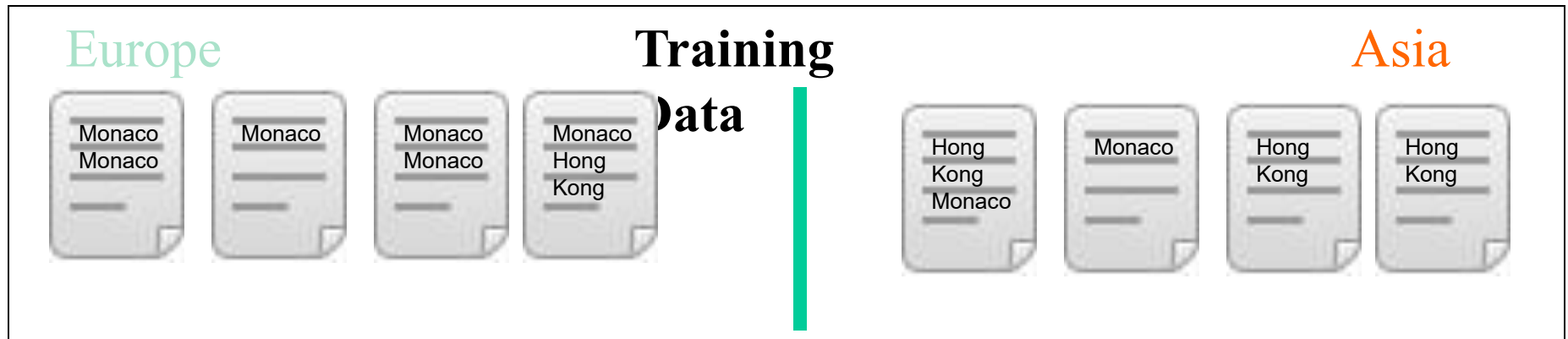
NB FACTORS:

- $P(A) = P(E) = \frac{1}{2}$
- $P(M|A) = \frac{1}{4}$
- $P(M|E) = \frac{3}{4}$

PREDICTIONS:

- $P(A, M) = \frac{1}{2} \times \frac{1}{4}$
- $P(E, M) = \frac{1}{2} \times \frac{3}{4}$
- $P(A|M) = \frac{1}{4}$
- $P(E|M) = \frac{3}{4}$

Text classification: Asia or Europe



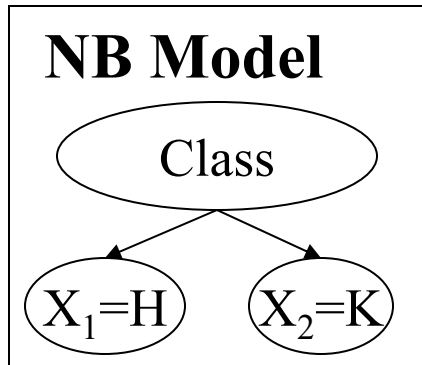
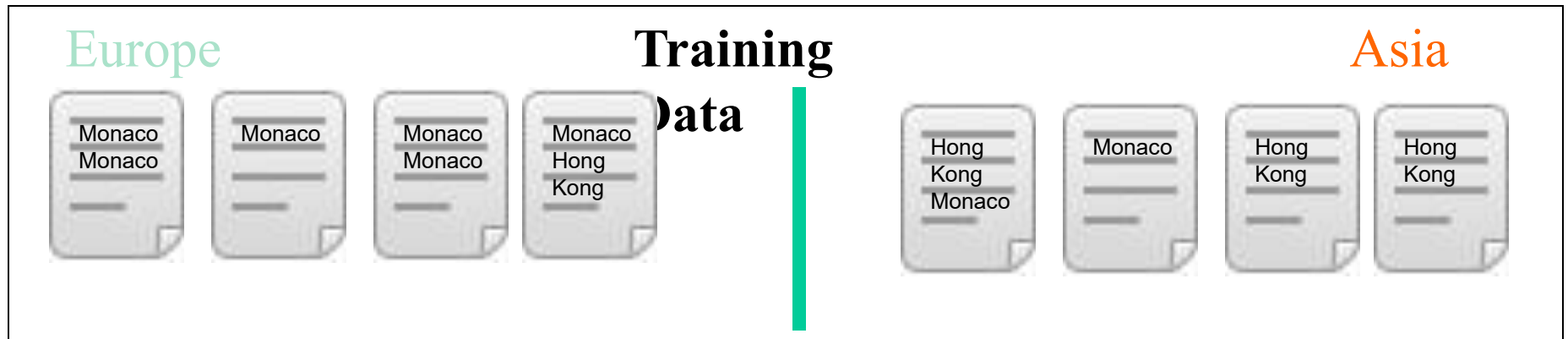
NB FACTORS:

- $P(A) = P(E) =$
- $P(H|A) = P(K|A) =$
- $P(H|E) = P(K|E) =$

PREDICTIONS:

- $P(A,H,K) =$
- $P(E,H,K) =$
- $P(A|H,K) =$
- $P(E|H,K) =$

Text classification: Asia or Europe



NB FACTORS:

- $P(A) = P(E) = \frac{1}{2}$
- $P(H|A) = P(K|A) = \frac{3}{5}$
- $P(H|E) = P(K|E) = \frac{1}{8}$

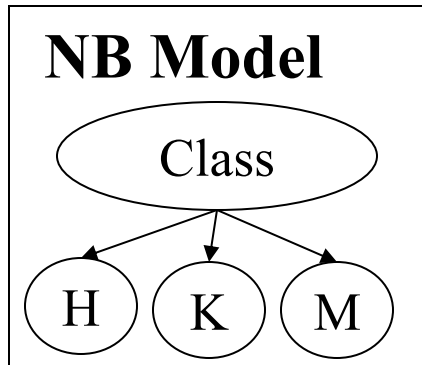
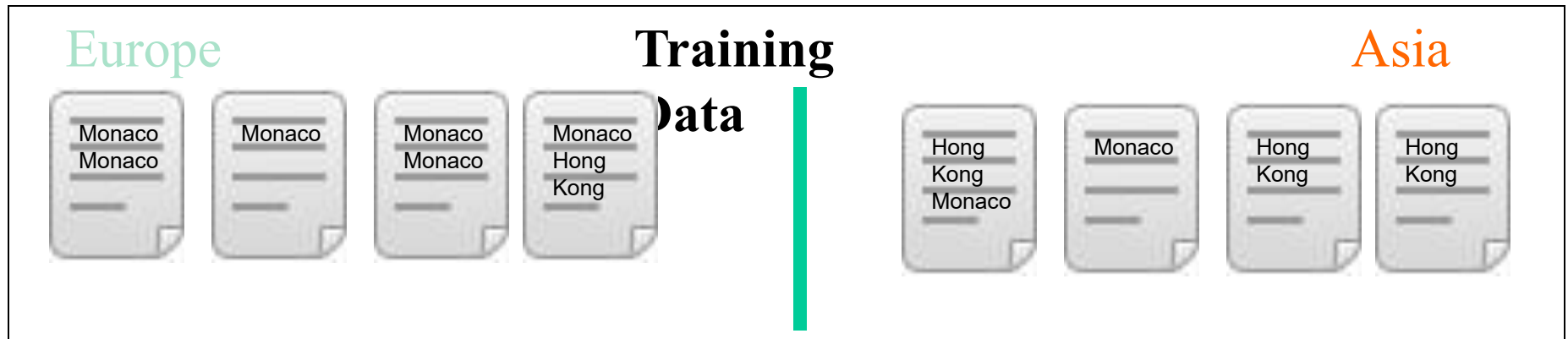
PREDICTIONS:

- $P(A,H,K) =$
- $P(E,H,K) =$
- $P(A|H,K) =$
- $P(E|H,K) =$

Handwritten calculations in red ink:

- $\frac{1}{2} \times \frac{3}{5} \times \frac{3}{5}$
- $\frac{1}{2} \times \frac{1}{8} \times \frac{1}{8}$
- $\frac{3}{10}$
- $\frac{1}{10}$

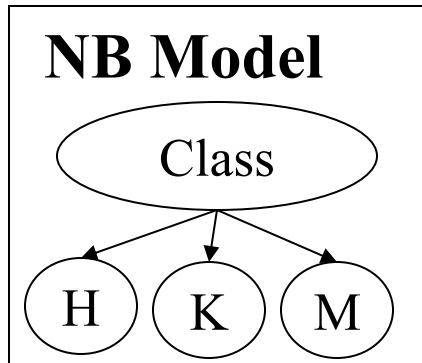
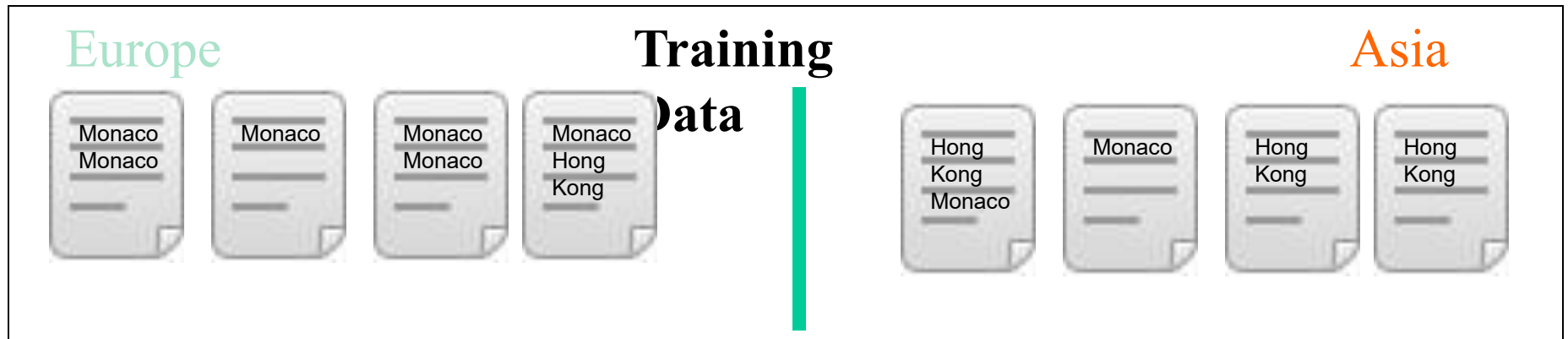
Text classification: Asia or Europe



NB FACTORS: PREDICTIONS:

- $P(A) = P(E) =$
- $P(M|A) =$
- $P(M|E) =$
- $P(H|A) = P(K|A) =$
- $P(H|E) = P(K|E) =$
- $P(A,H,K,M) =$
- $P(E,H,K,M) =$
- $P(A|H,K,M) =$
- $P(E|H,K,M) =$

Text classification: Asia or Europe



NB FACTORS: PREDICTIONS:

- $P(A) = P(E) = 1/2$
- $P(M|A) = 1/4$
- $P(M|E) = 3/4$
- $P(H|A) = P(K|A) = 3/8$
- $P(H|E) = P(K|E) = 1/8$
- $P(A, H, K, M) = \frac{1}{2} \times \frac{1}{4} \times \frac{3}{8} \times \frac{3}{8}$
- $P(E, H, K, M) = \frac{1}{2} \times \frac{3}{4} \times \frac{1}{4} \times \frac{1}{8}$
- $P(A|H, K, M) = \frac{3/8}{1/4}$
- $P(E|H, K, M) = \frac{1/8}{1/4}$



Generative vs Discriminative

Generative (Joint) Models

- Assume some functional form for $P(Y)$, $P(X|Y)$
- Estimate parameters of $P(X|Y)$, $P(Y)$ from training data
- Use Bayes rule to calculate $P(Y|X)$

Discriminative (Conditional) Models

- Assume some functional form for $P(Y|X)$
- Estimate parameters of $P(Y|X)$ directly from training data

Examples

- Generative: NB, Bayes Net, HMM
- Discriminative: Logistic Regression, SVM, (std) NN, CRF

Text Categorization with Word Features

(Zhang and Oles 2001)

- Features are presence of each **word** in a document and the document **class** (they do feature selection to use reliable indicator words)
- Tests on classic Reuters data set (and others)
 - Naïve Bayes: 77.0% F_1
 - Logistic regression: 86.4%
 - Support vector machine: 86.5%

Feature-Based Linear Classifiers

- Linear classifiers at classification time:
 - Linear function from feature sets $\{\phi_i\}$ to classes $\{y\}$.
 - Assign a weight w_i to each feature ϕ_i .
 - Feature is a function of input and output
- For a pair (x,y) , features vote with their weights:
 - $\text{vote}(y) = \sum w_i \phi_i(x,y)$
 - Choose the class y which maximizes $\sum w_i \phi_i(x,y)$
- We need probabilistic semantics to this method.
 - Log linear classifiers

Features for Multi-Class Problems

- $\phi_i(x,y) = 1$ if $\phi_i(x) = 1$ and $\text{label}(x) = y$
= 0 otherwise

Assign a weight for each feature $\phi_i(x,y)$, i.e., a different weight for each prediction y

For a pair (x,y) , features vote with their weights:

- $\text{vote}(y) = \sum w_i \phi_i(x,y)$
- **Choose the class y which maximizes $\sum w_i \phi_i(x,y)$**
- This can be written in linear algebra notation as $W^T X$ and it will yield a $|X| \times |Y|$ matrix with a score for each (x,y)

Exponential Models

(log-linear, maxent, Logistic, Gibbs)

- **Model:** use the scores as probabilities:

$$p(y|x; w) = \frac{\exp(w \cdot \phi(x, y))}{\sum_{y'} \exp(w \cdot \phi(x, y'))}$$

← Make positive
← Normalize

- **Learning:** maximize the (log) conditional likelihood of training data

$$\{(x_i, y_i)\}_{i=1}^n$$

$$L(w) = \sum_{i=1}^n \log p(y_i|x_i; w) \quad w^* = \arg \max_w L(w)$$

- **Parameters:** w
- **Prediction:** output $\operatorname{argmax}_y p(y|x; w)$

“all models are wrong
some are useful!”

-- *George Box*

Derivative of Log-linear Model

$$p(y|x; w) = \frac{\exp(w \cdot \phi(x, y))}{\sum_{y'} \exp(w \cdot \phi(x, y'))}$$

- Unfortunately, $\operatorname{argmax}_w L(w)$ doesn't have a close formed solution
- We will have to differentiate and use gradient ascent

$$L(w) = \sum_{i=1}^n \log p(y_i|x_i; w)$$

$$L(w) = \sum_{i=1}^n \left(w \cdot \phi(x_i, y_i) - \log \sum_y \exp(w \cdot \phi(x_i, y)) \right)$$

$$\frac{\partial L(w)}{\partial w_{jk}} = \sum_{i=1}^n \left(\phi_{jk}(x_i, y_i) - p(k|x_i; w) \phi_{jk}(x_i, k) \right)$$

Total count of feature j in
candidates with class k

Expected count of
feature j in predicted
candidates of class k

Proof

(Conditional Likelihood Derivative)

- Recall

$$p(y|x; w) = \frac{\exp(w \cdot \phi(x, y))}{\sum_{y'} \exp(w \cdot \phi(x, y'))}$$

$$P(Y|X, w) = \prod_{(x,y) \in D} p(y|x, w)$$

- We can separate this into two components:

$$\log P(Y|X, w) = \sum_{i=1}^n (w \cdot \phi(x_i, y_i)) - \sum_{i=1}^n \left(\log \sum_y \exp(w \cdot \phi(x_i, y)) \right)$$

- The derivative is the difference between the derivatives of each component

$$\log P(Y|X, w) = N(w) - D(w)$$

Proof: Numerator

$$\begin{aligned}\frac{\partial N(w)}{\partial w_{jk}} &= \frac{\partial \sum_{i=1}^n (\sum_l (w_{ly_i} \varphi_{ly_i}(x_i, y_i)))}{\partial w_{jk}} \\ &= \sum_{i=1}^n \frac{\partial (\sum_l (w_{ly_i} \varphi_{ly_i}(x_i, y_i)))}{\partial w_{jk}} \\ &= \sum_{i=1}^n \varphi_{jk}(x_i, y_i)\end{aligned}$$

Derivative of the numerator is:

the empirical count of feature j with class k

Note: $\varphi_{jk}(x_i, y_i) = 0$ if $y \neq k$

Proof: Denominator

$$\begin{aligned}\frac{\partial D(w)}{\partial w_{jk}} &= \frac{\partial \sum_{i=1}^n \log \sum_y \exp(\sum_l (w_{ly} \phi_{ly}(x_i, y)))}{\partial w_{jk}} \\ &= \sum_{i=1}^n \frac{1}{\sum_{y'} \exp(\sum_l (w_{ly'} \phi_{ly'}(x_i, y')))} \frac{\partial \sum_y \exp(\sum_l (w_{ly} \phi_{ly}(x_i, y)))}{\partial w_{jk}} \\ &= \sum_{i=1}^n \frac{1}{\sum_{y'} \exp(\sum_l (w_{ly'} \phi_{ly'}(x_i, y')))} \sum_y \frac{\exp(\sum_l (w_{ly} \phi_{ly}(x_i, y)))}{1} \frac{\partial \sum_l (w_{ly} \phi_{ly}(x_i, y))}{\partial w_{jk}} \\ &= \sum_{i=1}^n \sum_y \frac{\exp(\sum_l (w_{ly} \phi_{ly}(x_i, y)))}{\sum_{y'} \exp(\sum_l (w_{ly'} \phi_{ly'}(x_i, y')))} \phi_{jk}(x_i, y) \\ &= \sum_{i=1}^n \sum_y P(y|x_i; w) \phi_{jk}(x_i, y) \\ &= \sum_{i=1}^n p(k|x_i; w) \phi_{jk}(x_i, k) \quad = \text{expected count of} \\ & \quad \text{feature } j \text{ predicted with class } k\end{aligned}$$

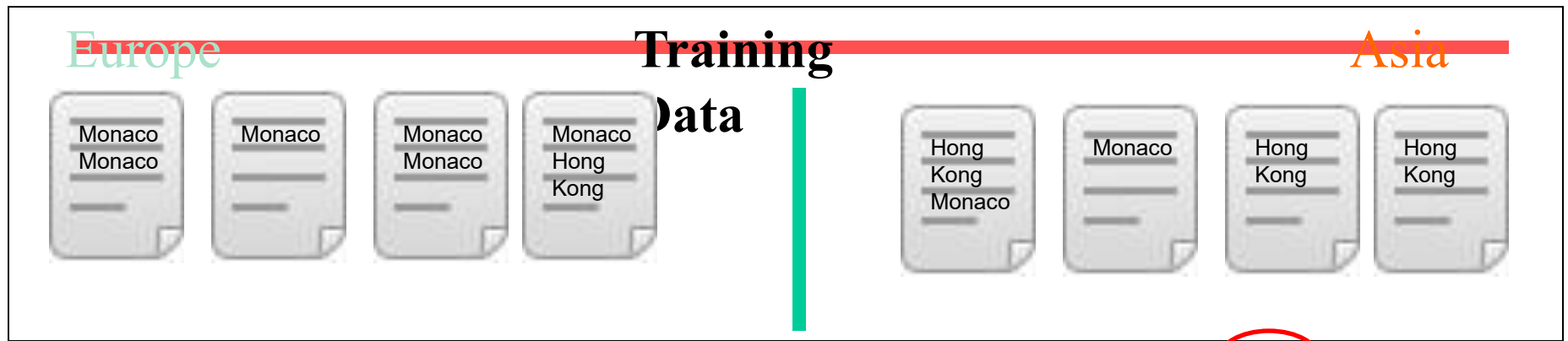
Proof (concluded)

$$\frac{\partial P(Y|X; w)}{\partial w_{jk}} = \text{actualcount}(\phi_{jk}) - \text{predictedcount}(\phi_{jk})$$

- The optimum parameters are the ones for which each feature's **predicted expectation** equals its **empirical expectation**. The optimum distribution is:
 - Always unique (but parameters may not be unique)
 - Always exists (if feature counts are from actual data).
- These models are also called maximum entropy models because we find the model has the maximum entropy while satisfying the constraints:

$$E_p(\phi_i) = E_{\tilde{p}}(\phi_i), \forall i$$

Text classification: Asia or Europe

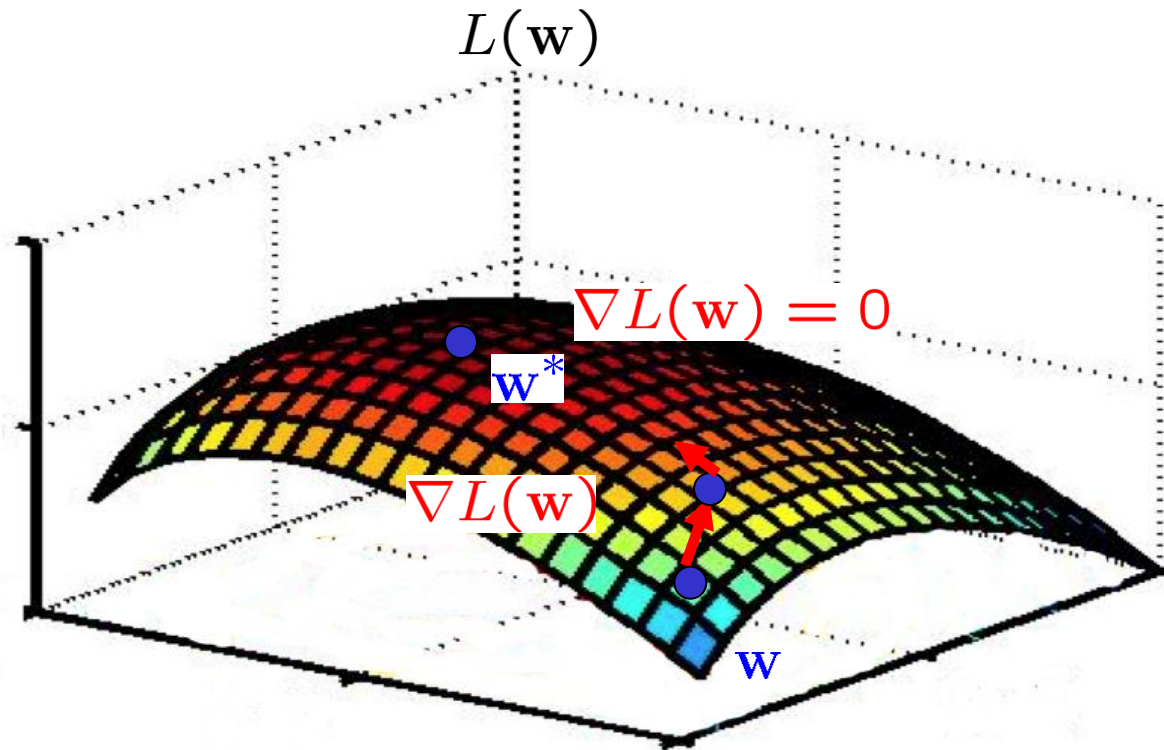


$$\begin{aligned}
 P(Y | X) &= \frac{e^{2me} + e^{2ma}}{e^{ha+ka+ma}} \cdot \frac{e^{me} + e^{ma}}{e^{ma}} \cdot \frac{e^{2me} + e^{2ma}}{e^{ma}} \cdot \frac{e^{me+he+ke} + e^{ma+ha+ka}}{e^{ha+ka}} \\
 P(Y | X) &= \frac{e^{2me} + e^{2ma}}{e^{hka+ma}} \cdot \frac{e^{me} + e^{ma}}{e^{ma}} \cdot \frac{e^{2me} + e^{2ma}}{e^{hka}} \cdot \frac{e^{me+hke} + e^{ma+hka}}{e^{hka}}
 \end{aligned}$$

Note: In the first equation, terms like $ha+ka$, $he+ke$, $me+he+ke$, and $ma+ha+ka$ are circled in red to show their equivalence to hka and hke in the second equation.

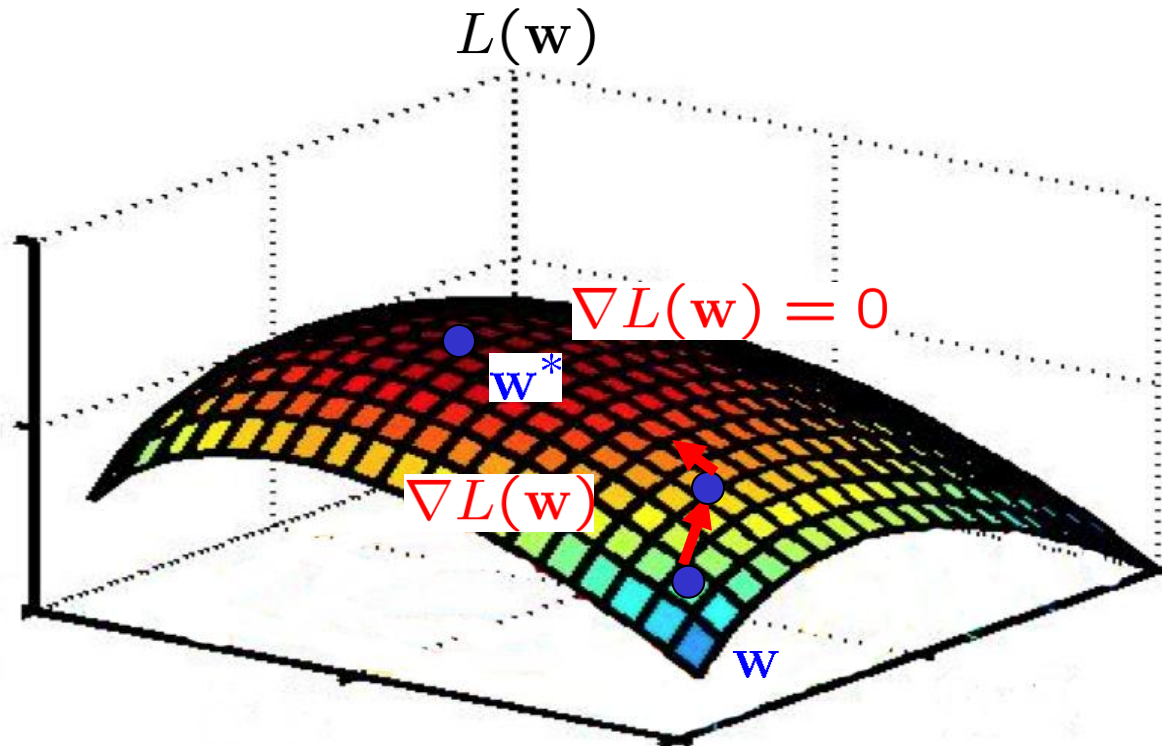
Both equations are the same: $ha+ka=hka$; $he+ke=hke$... will have same optima

Unconstrained Optimization



- Basic idea: move uphill from current guess
- Gradient ascent / descent follows the gradient incrementally
- At local optimum, derivative vector is zero
- Will converge if step sizes are small enough, but not efficient
- All we need is to be able to evaluate the function and its derivative

Unconstrained Optimization



- For convex functions, a local optimum will be global
- Stochastic Gradient Descent / Batch Stochastic Gradient Descent
- Basic gradient descent isn't very efficient, but there are simple enhancements which take into account previous gradients: RMSProp, ADAM, AdaGrad, AdamW, etc.

What About Overfitting?

- For Naïve Bayes, we were worried about zero counts in MLE estimates
 - Can that happen here?
- Regularization (smoothing) for Log-linear models
 - Instead, we worry about large feature weights
 - Add a regularization term to the likelihood to push weights towards zero

$$L(w) = \sum_{i=1}^n \log p(y_i | x_i; w) - \frac{\lambda}{2} \|w\|^2$$

Derivative for Regularized Maximum Entropy

- Unfortunately, $\operatorname{argmax}_w L(w)$ still doesn't have a close formed solution
- We will have to differentiate and use gradient ascent

$$L(w) = \sum_{i=1}^n \left(w \cdot \phi(x_i, y_i) - \log \sum_y \exp(w \cdot \phi(x_i, y)) \right) - \frac{\lambda}{2} \|w\|^2$$

$$\frac{\partial}{\partial w_j} L(w) = \sum_{i=1}^n \left(\phi_j(x_i, y_i) - \sum_y p(y|x_i; w) \phi_j(x_i, y) \right) - \lambda w_j$$

Total count of feature j
in correct candidates

Expected count of
feature j in predicted
candidates

Big weights
are bad

L1 and L2 Regularization

L2 Regularization for Log-linear models

- Instead, we worry about large feature weights
- Add a regularization term to the likelihood to push weights towards zero

$$L(w) = \sum_{i=1}^n \log p(y_i | x_i; w) - \frac{\lambda}{2} \|w\|^2$$

Regularization Constant

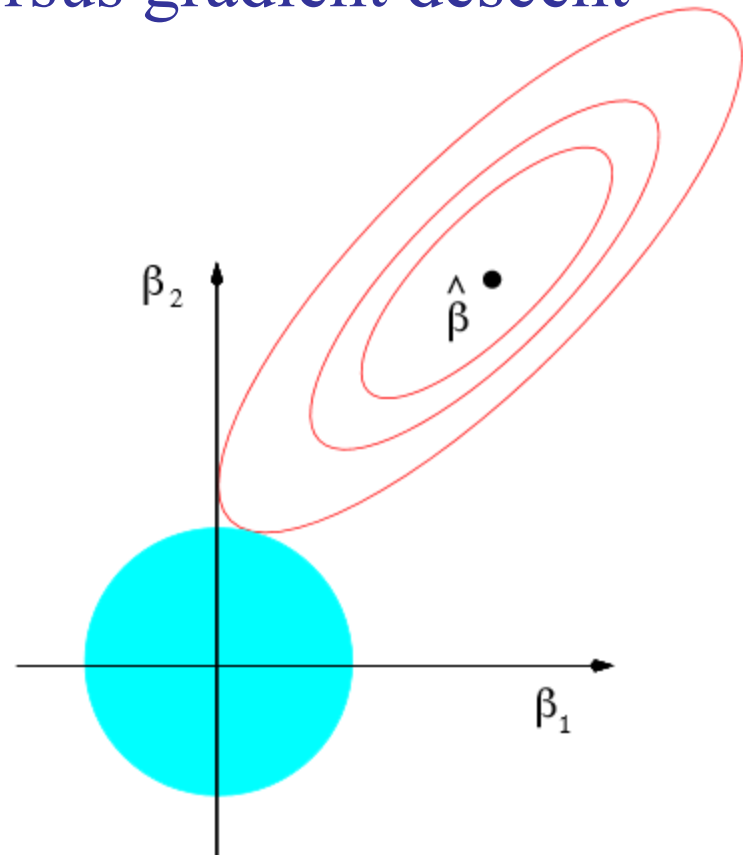
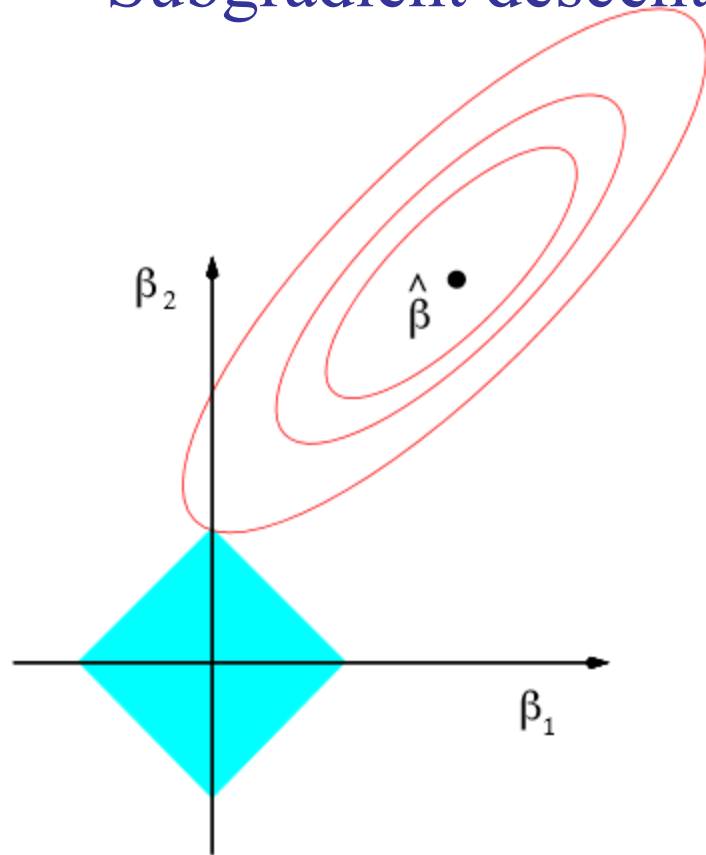
L1 Regularization for Log-linear models

- Instead, we worry about number of active features
- Add a regularization term to the likelihood to push weights to zero
- For L1 regularization, we need to compute subgradients.

$$L(w) = \sum_{i=1}^n \log p(y_i | x_i; w) - \lambda \|w\|$$

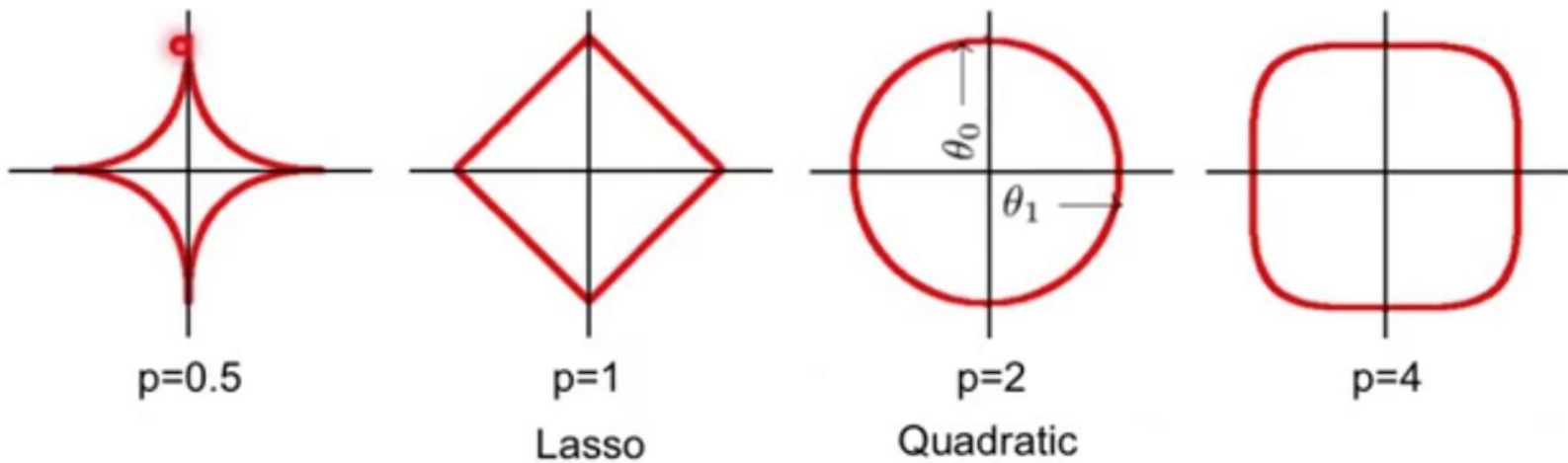
L1 vs L2

- Optimizing L1 harder
 - Discontinuous objective function
 - Subgradient descent versus gradient descent

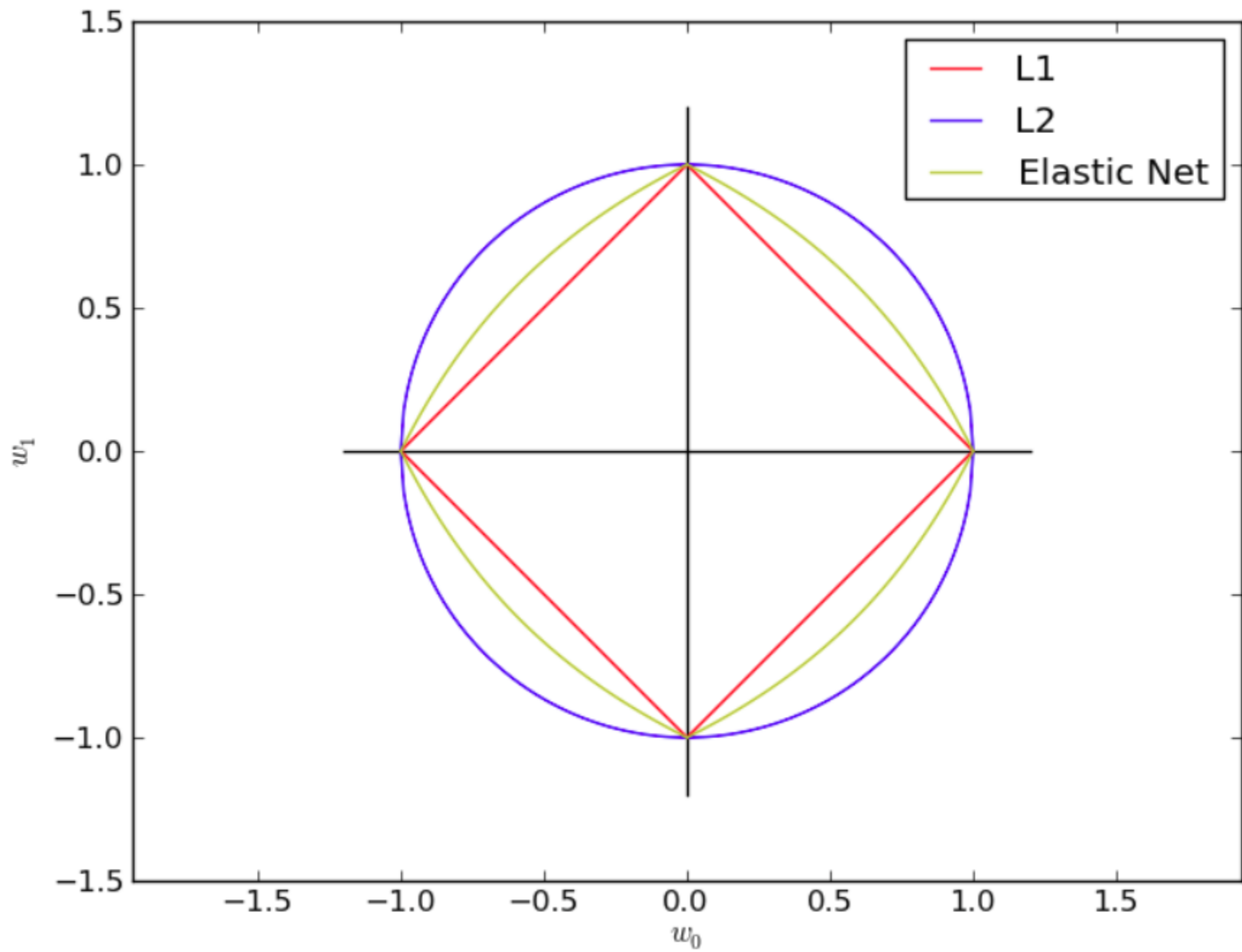


L_p Norms for Regularization

Isosurfaces: $\|\theta\|_p = \text{constant}$



L_0 = limit as $p \rightarrow 0$: "number of nonzero weights", a natural notion of complexity



How to pick weights?

- Goal: choose “best” vector w given training data
 - For now, we mean “best for classification”
- The ideal: the weights which have greatest test set accuracy / F1 / whatever
 - But, don’ t have the test set
 - Must compute weights from training set
- Maybe we want weights which give best training set accuracy?
 - May not (does not) generalize to test set
 - Easy to overfit
- Use devset

Diving Deeper into Feature Engineering

Construct Better Features

- Key to machine learning is having good features
- In gen 2 ML, large effort devoted to constructing appropriate features
- Ideas??

Issues in document representation

Cooper's concordance of Wordsworth was published in 1911. The applications of full-text retrieval are legion: they include résumé scanning, litigation support and searching published journals on-line.

- *Cooper's vs. Cooper vs. Coopers.*
- *Full-text vs. full text vs. {full, text} vs. fulltext.*
- *résumé vs. resume.*

Punctuation

- *Ne'er*: use language-specific, handcrafted “locale” to normalize.
- *State-of-the-art*: break up hyphenated sequence.
- *U.S.A.* vs. *USA*
- *a.out*

Numbers

- 3/12/91
- Mar. 12, 1991
- 55 B.C.
- B-52
- 100.2.86.144
 - Generally, don't represent as text
 - Creation dates for docs

Possible Feature Ideas

- Look at capitalization (may indicated a proper noun)
- Look for commonly occurring sequences
 - E.g. New York, New York City
 - Limit to 2-3 consecutive words
 - Keep all that meet minimum threshold (e.g. occur at least 5 or 10 times in corpus)

Case folding

- Reduce all letters to lower case
- Exception: upper case in mid-sentence
 - *e.g., General Motors*
 - *Fed vs. fed*
 - *SAIL vs. sail*

Thesauri and Soundex

- Handle synonyms and spelling variations
 - Hand-constructed equivalence classes
 - e.g., *car* = *automobile*

Spell Correction

- Look for all words within (say) edit distance 3 (Insert/Delete/Replace) at query time
 - *e.g., artificial intelligence*
- Spell correction is expensive and slows the processing significantly
 - Invoke only when index returns zero matches?

Stemming

- Are there different index terms?
 - retrieve, retrieving, retrieval, retrieved, retrieves...
- Stemming algorithm:
 - (retrieve, retrieving, retrieval, retrieved, retrieves) ⇒ **retriev**
 - Strips prefixes of suffixes (-s, -ed, -ly, -ness)
 - Morphological stemming
- Problems: sand / sander & wand / wander

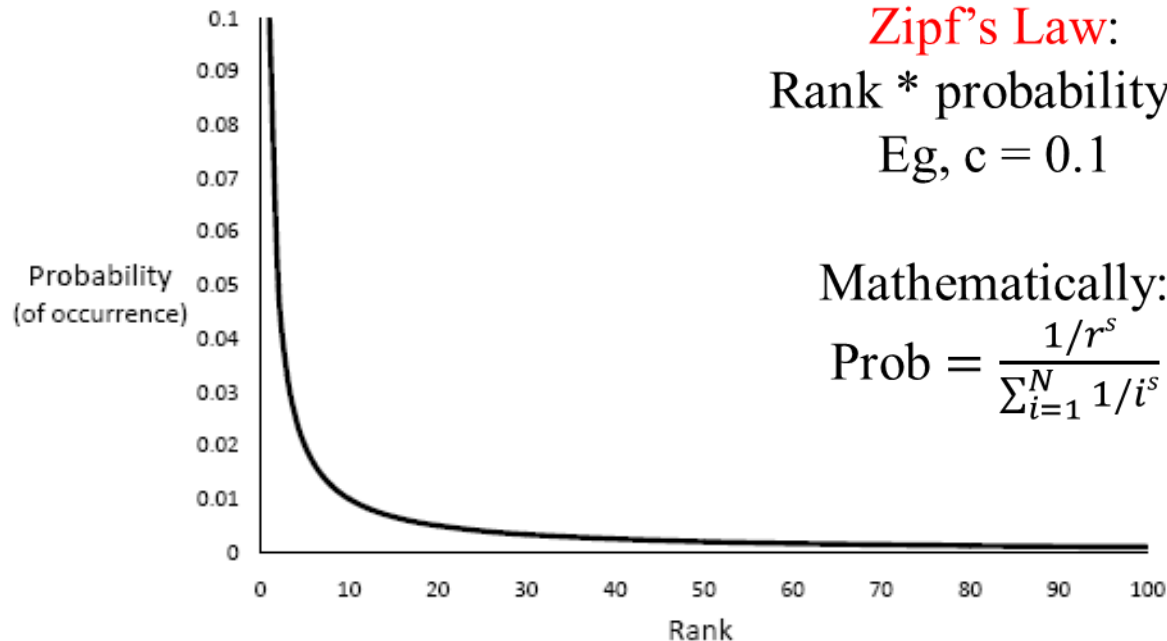
Features

Sec 15.3.2

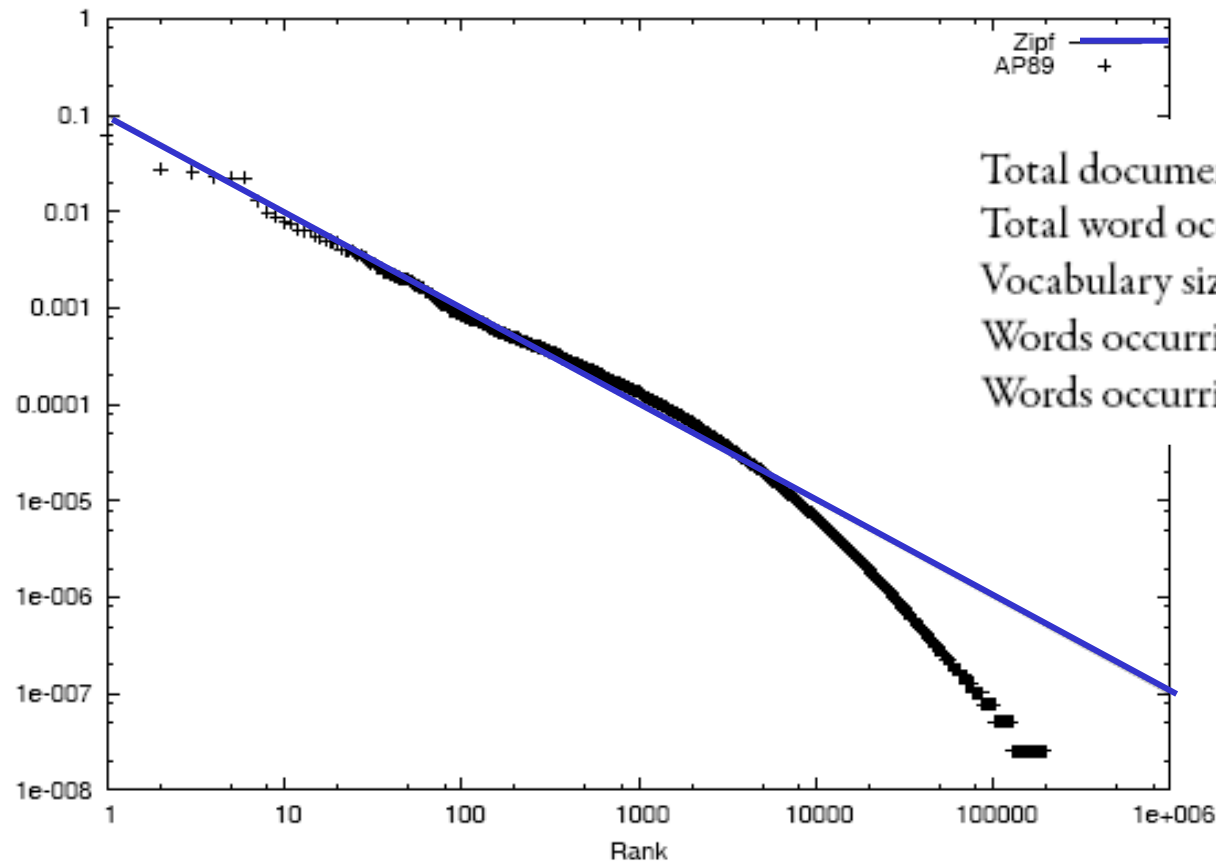
-
- Domain-specific features and weights: *very* important in real performance
 - Upweighting: Counting a word as if it occurred twice:
 - title words (Cohen & Singer 1996)
 - first sentence of each paragraph (Murata, 1999)
 - In sentences that contain title words (Ko *et al*, 2002)

Properties of Text

- Word frequencies - skewed distribution
- 'The' and 'of' account for 10% of all words
- Six most common words account for 40%



Associate Press Corpus `AP89`



Total documents	84,678
Total word occurrences	39,749,179
Vocabulary size	198,763
Words occurring > 1000 times	4,169
Words occurring once	70,064

Middle Ground

- Very common words → bad features

- Language-based stop list:

words that bear little meaning

20-500 words

http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words

- Subject-dependent stop lists

- Very rare words *also* bad features

Drop words appearing less than k times / corpus

Word Frequency

- Which word is more indicative of document similarity?
 - ‘book,’ or ‘Rumpelstiltskin’?
 - Need to consider “**document frequency**”--- how frequently the word appears in doc collection.
- Which doc is a better match for the query “Kangaroo”?
 - One with a single mention of Kangaroos... or a doc that mentions it 10 times?
 - Need to consider “**term frequency**”--- how many times the word appears in the current document.

TF x IDF

$$w_{ik} = tf_{ik} * \log(N / n_k)$$

T_k = term k in document D_i

tf_{ik} = frequency of term T_k in document D_i

idf_k = inverse document frequency of term T_k in C

$$idf_k = \log\left(\frac{N}{n_k}\right)$$

N = total number of documents in the collection C

n_k = the number of documents in C that contain T_k

Inverse Document Frequency

- IDF provides high values for rare words and low values for common words

$$\log\left(\frac{10000}{10000}\right) = 0$$

$$\log\left(\frac{10000}{5000}\right) = 0.301$$

$$\log\left(\frac{10000}{20}\right) = 2.698$$

$$\log\left(\frac{10000}{1}\right) = 4$$

- Add 1 to avoid 0.

TF-IDF normalization

- Normalize the values
 - so longer docs not given more value (fairness)
 - force all values to fall within a certain range: [0, 1]

$$w_{ik} = \frac{tf_{ik} (1 + \log(N / n_k))}{\sqrt{\sum_{k=1}^t (tf_{ik})^2 [1 + \log(N / n_k)]^2}}$$

Use of TFIDF in LR

- Earlier:
 - Presence of feature is a value $\{0, 1\}$
 - Count of feature can be a value $\{0, 1, 2, \dots\}$
- Now: Use TFIDF as feature value
 - Weights are still learned as is