CS 671 NLP
MACHINE LEARNING

Reading

- Christopher M. Bishop, Pattern recognition and machine learning. Springer, 2006.

Learning in NLP

- **Language models may be Implicit** : we can’t describe how we use language so effortlessly

- **Unknown future cases:** Constantly need to interpret sentences we have never heard before

- **Model structures**: Learning can reveal properties (regularities) of the language system
  - Latent structures / Dimensionality reduction : **reduce complexity** and improve performance

Feedback in Learning

- Type of feedback:
  - Supervised learning: correct answers for each example
    - Discrete (categories) : classification
    - Continuous : regression
  - Unsupervised learning: correct answers not given
  - Reinforcement learning: occasional rewards

Inductive learning

Simplest form: learn a function from examples

An example is a pair (x, y) : x = data, y = outcome

assume: y drawn from function f(x) : y = f(x) + noise

f = target function

Problem: find a hypothesis h

such that $h \approx f$

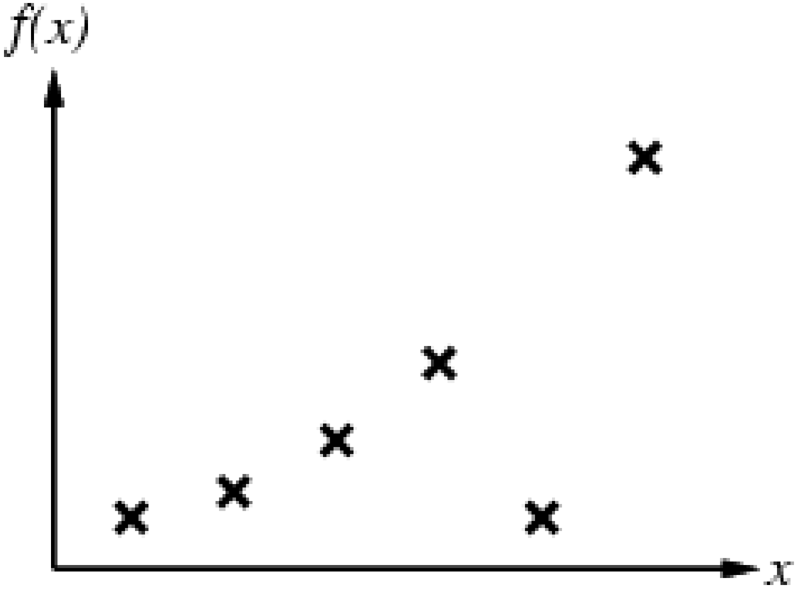
given a training set of examples

Note: highly simplified model :

- Ignores prior knowledge : some h may be more likely
- Assumes lots of examples are available
- Objective: maximize prediction for unseen data – Q. How?

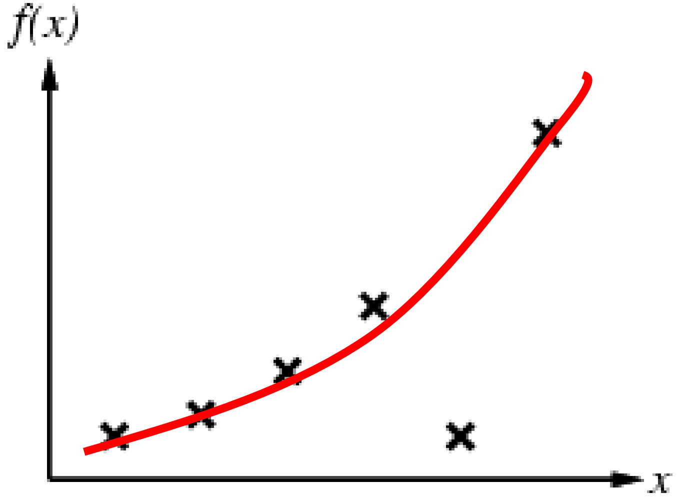
Inductive learning method

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)
- E.g., curve fitting:



Regression vs Classification

y = f(x)



Regression:

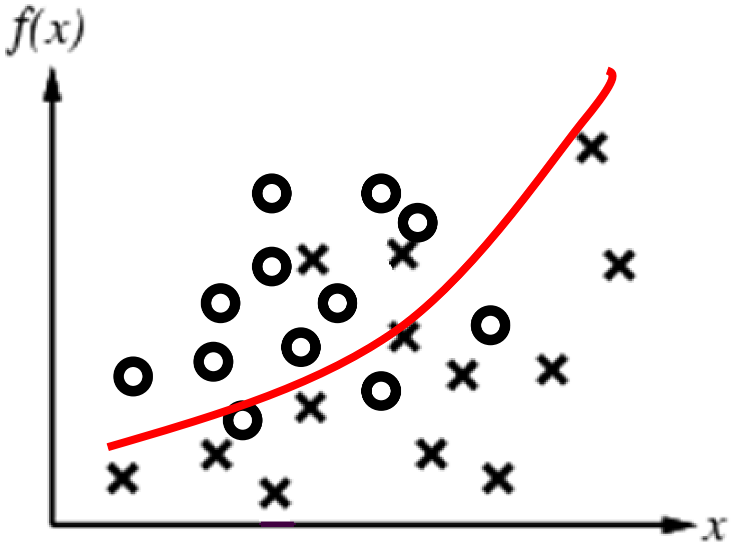
y is continuous

Classification:

y : set of discrete values

e.g. classes C_1, C_2, C_3...

y ∈ {1,2,3...}



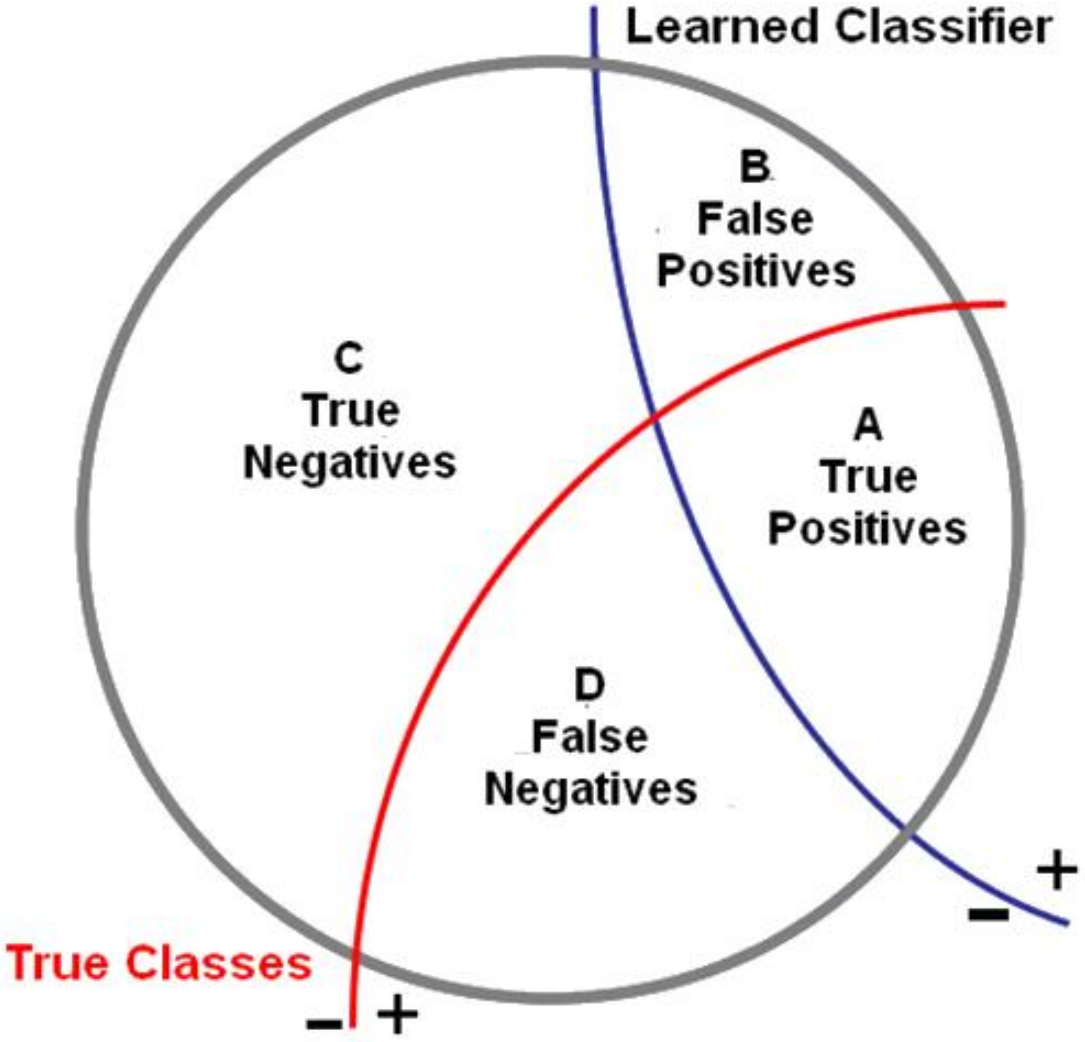
Precision vs Recall

Precision:

A / Retrieved Positives

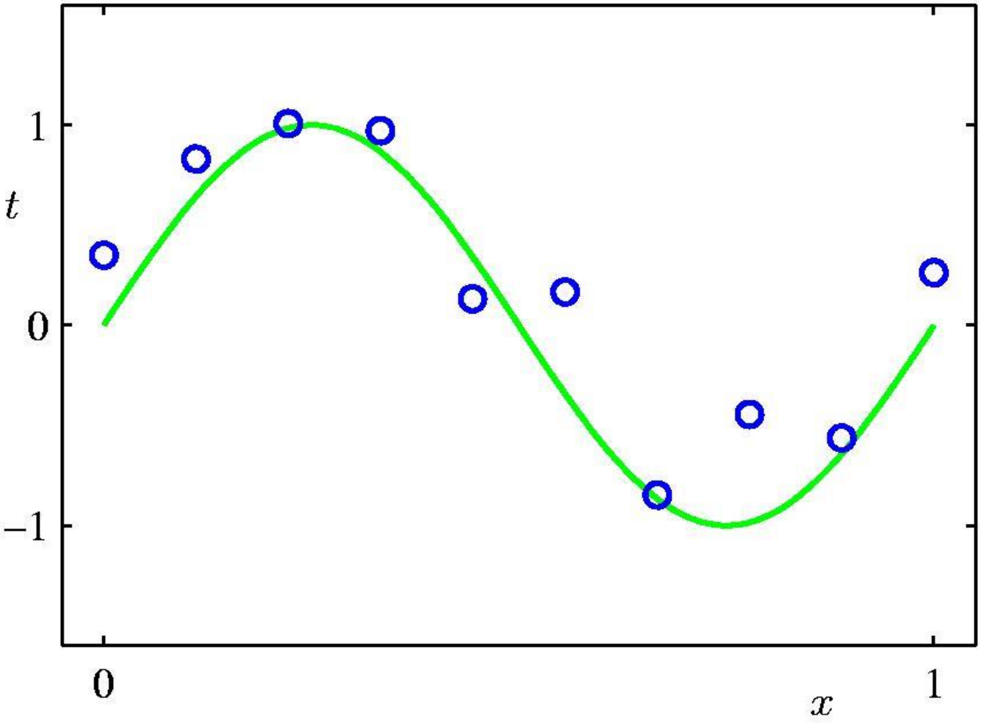
Recall:

A / Actual Positives



Regression

Polynomial Curve Fitting



$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Linear Regression

$$\mathbf{y = f(x) = \Sigma_i\, w_i\, .\, \phi_i(x)}$$

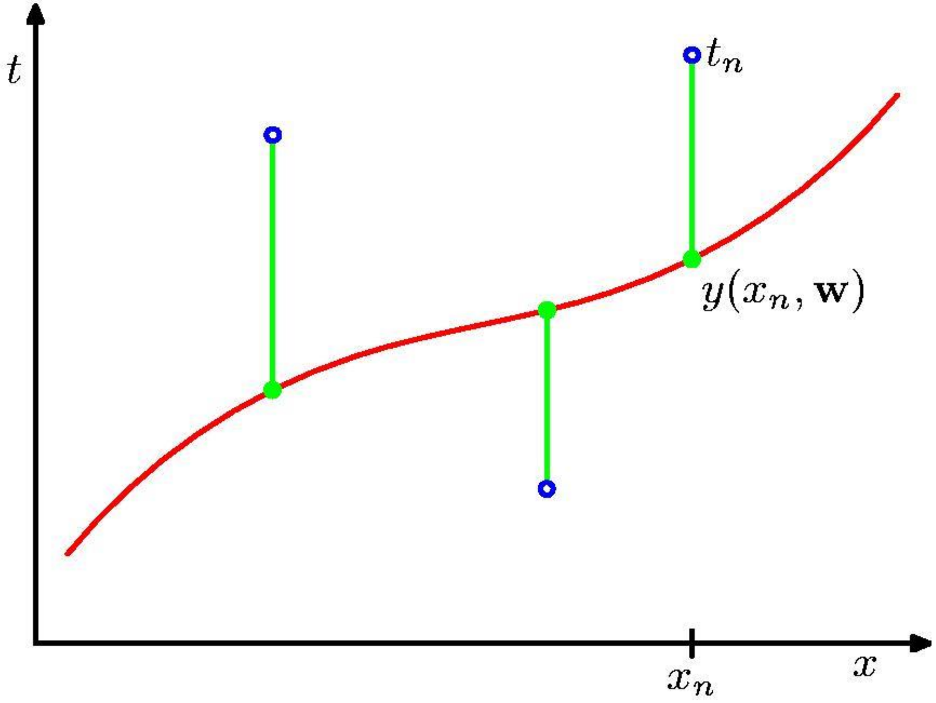
$\boldsymbol{\phi_i(x)}$: basis function

w_i : weights

Linear : function is linear in the weights

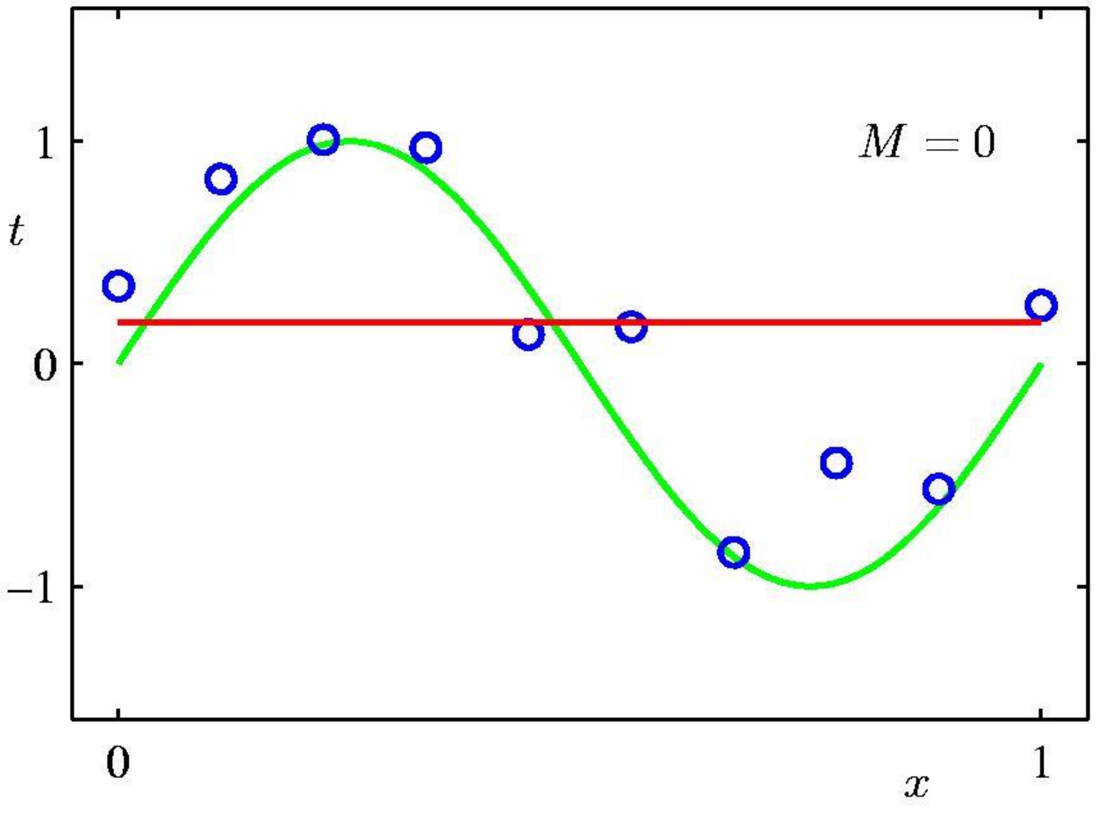
Quadratic error function --> derivative is linear in **w**

Sum-of-Squares Error Function

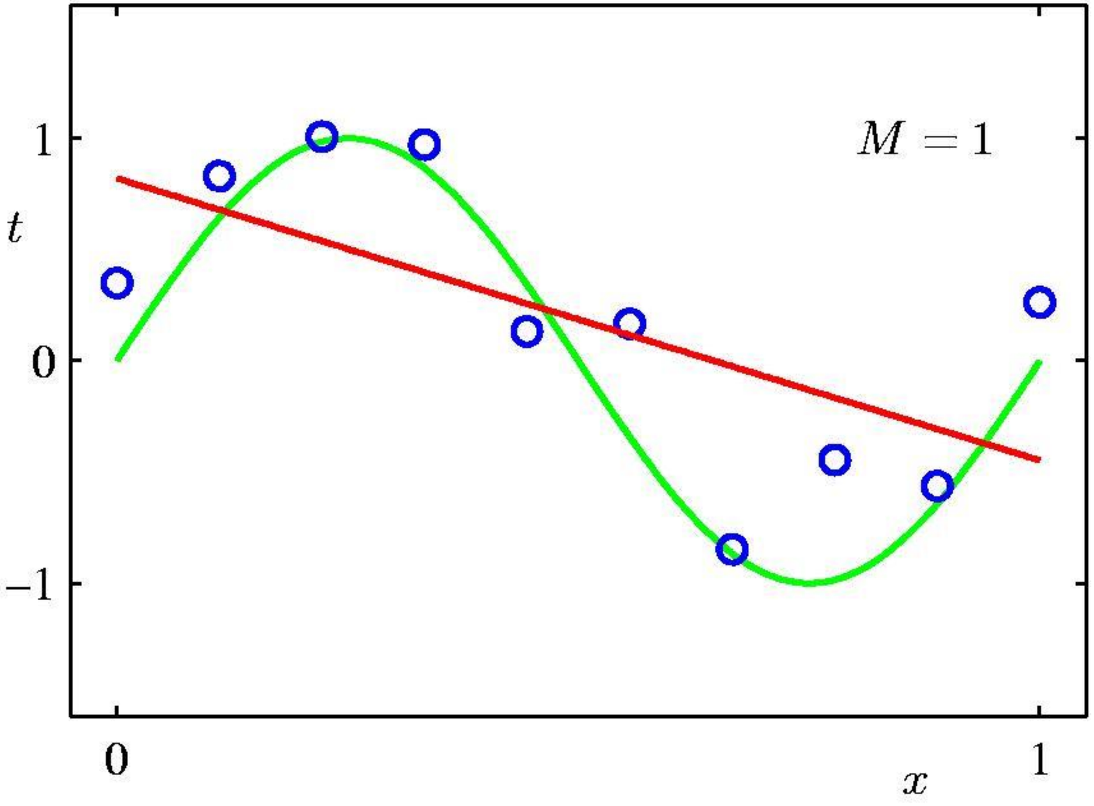


$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2$$

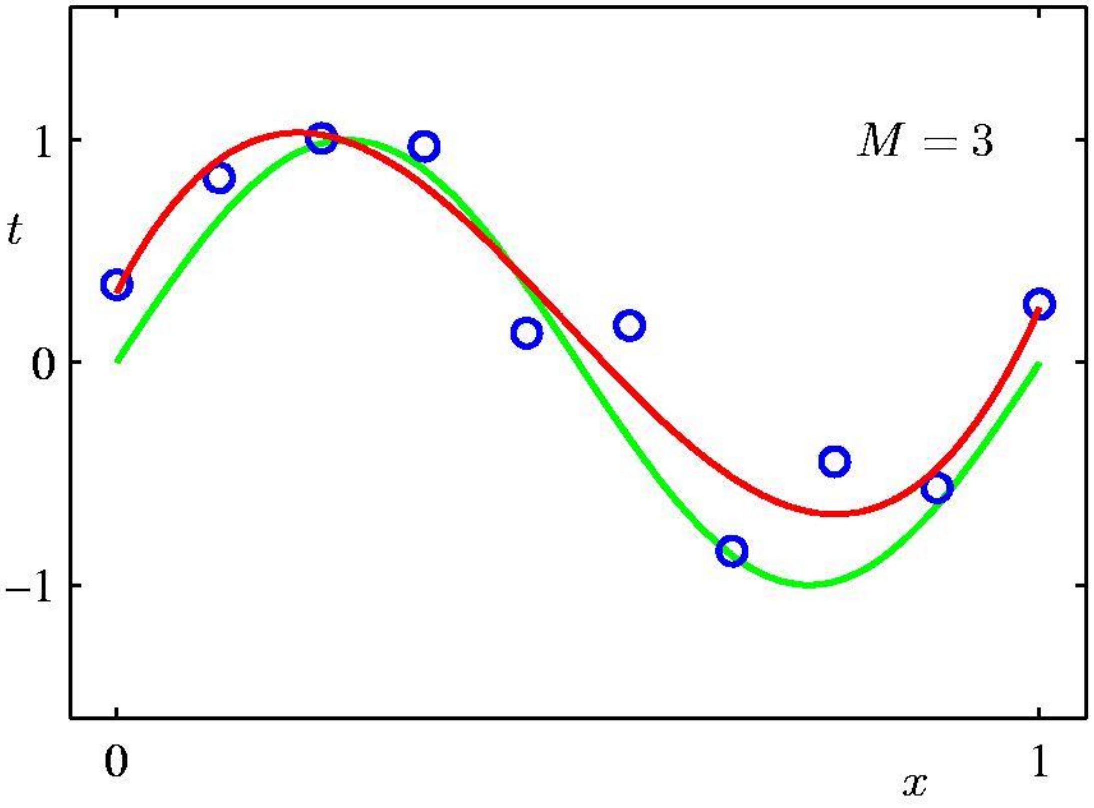
0^{th} Order Polynomial



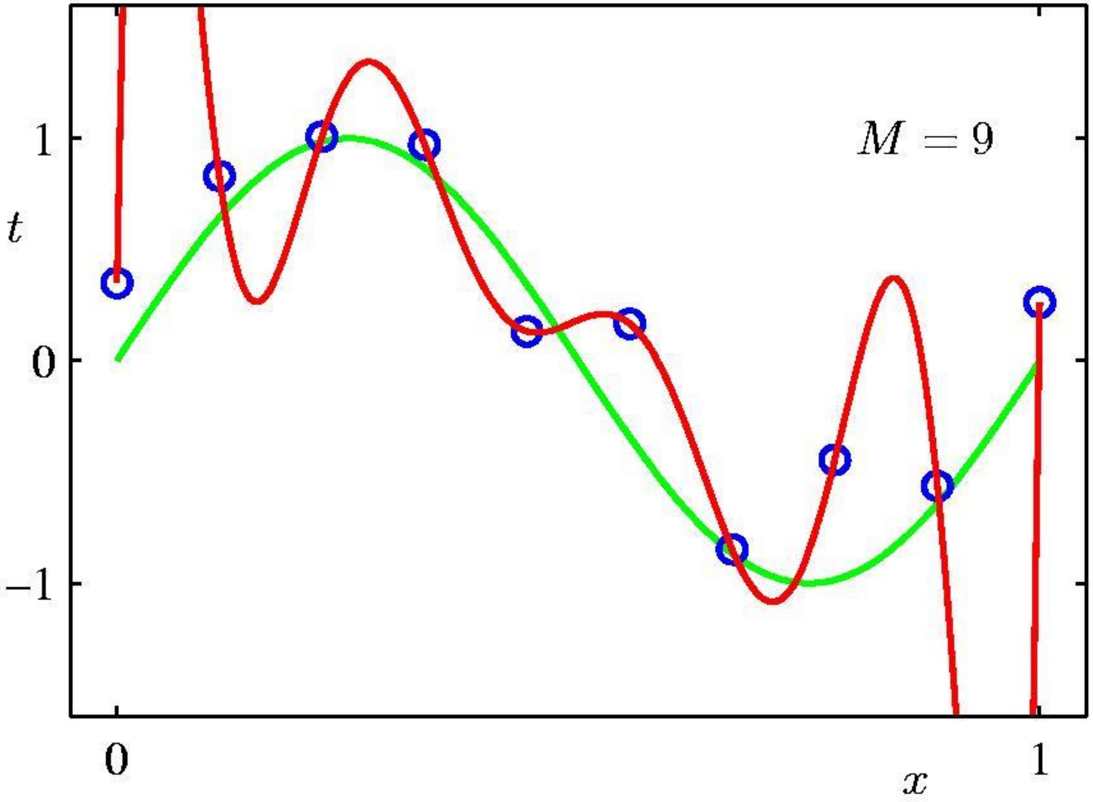
1st Order Polynomial



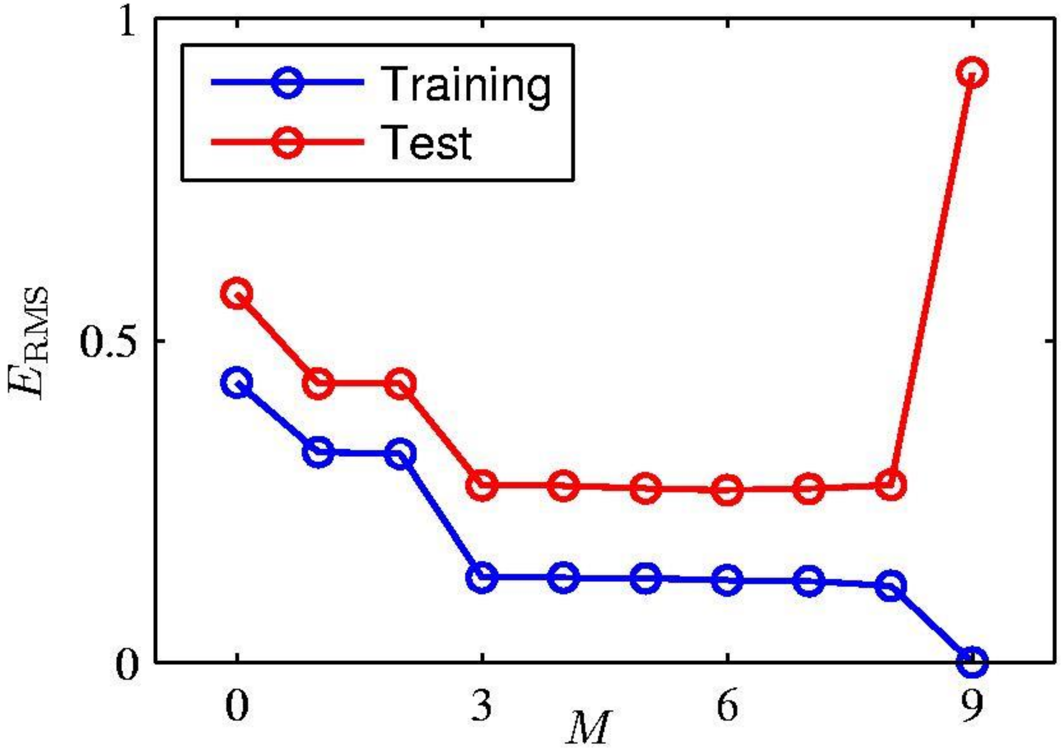
3^{rd} Order Polynomial



9th Order Polynomial



Over-fitting

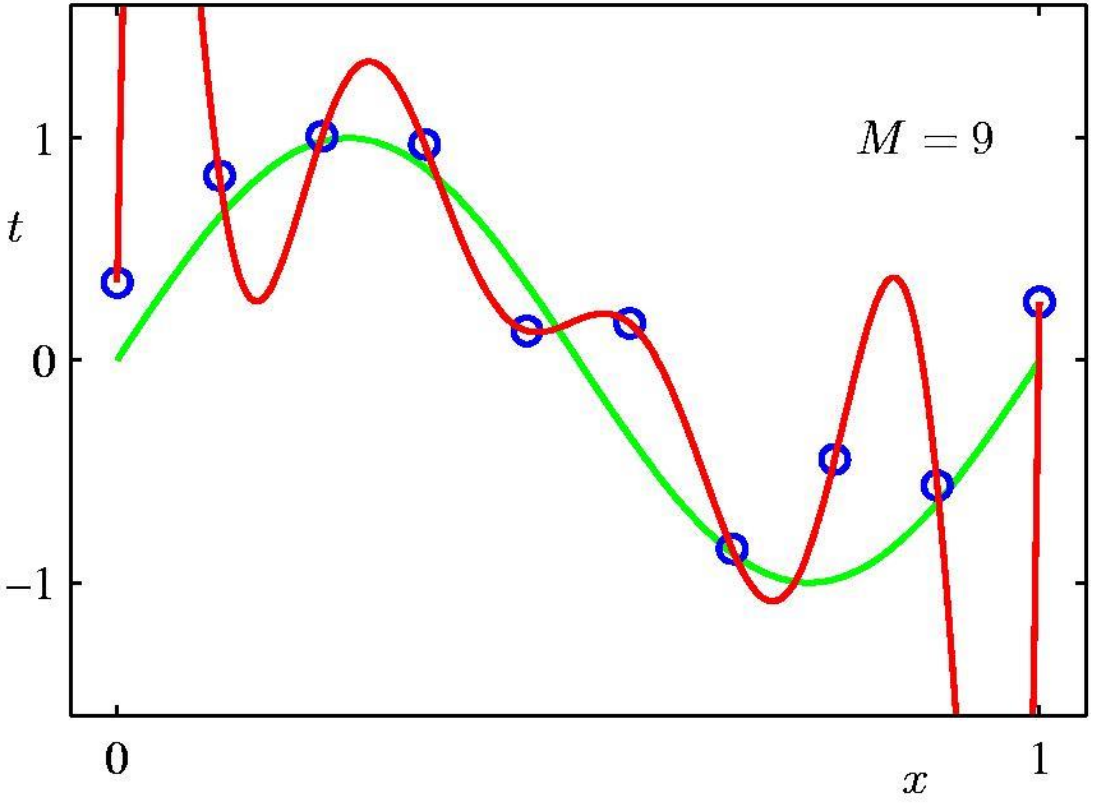


Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^{\star})/N}$

Polynomial Coefficients

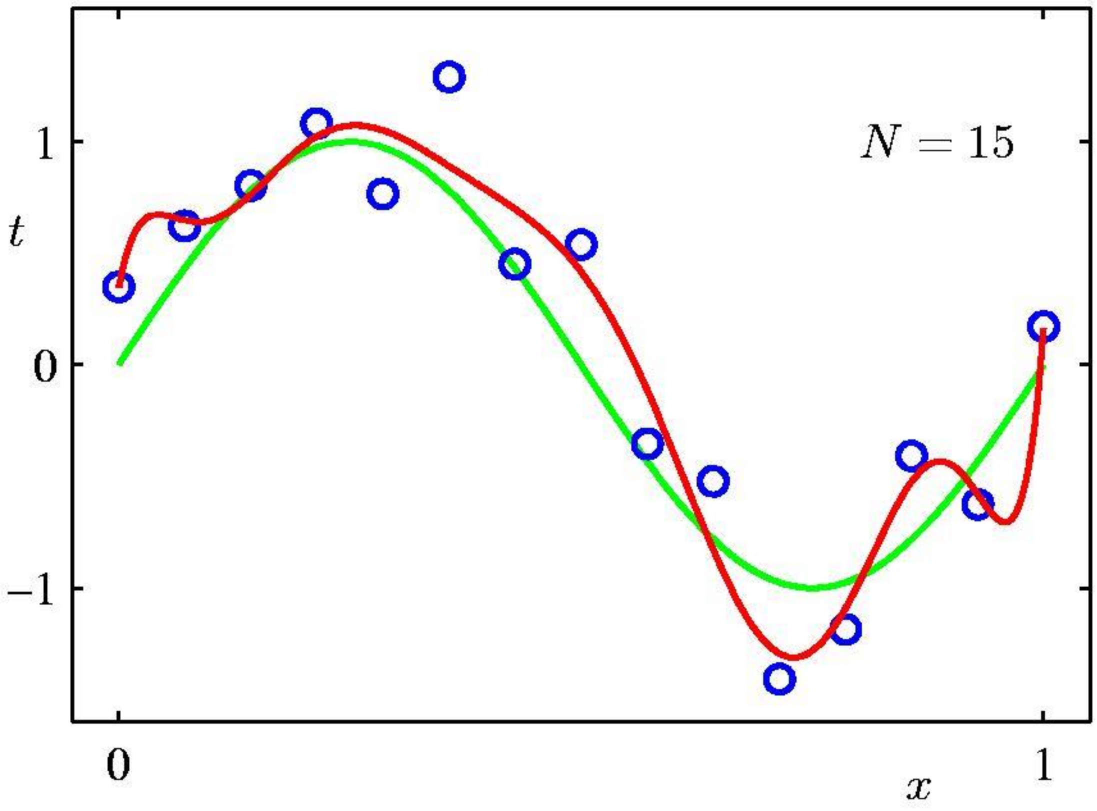
	$M = 0$	$M = 1$	$M = 3$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

9th Order Polynomial



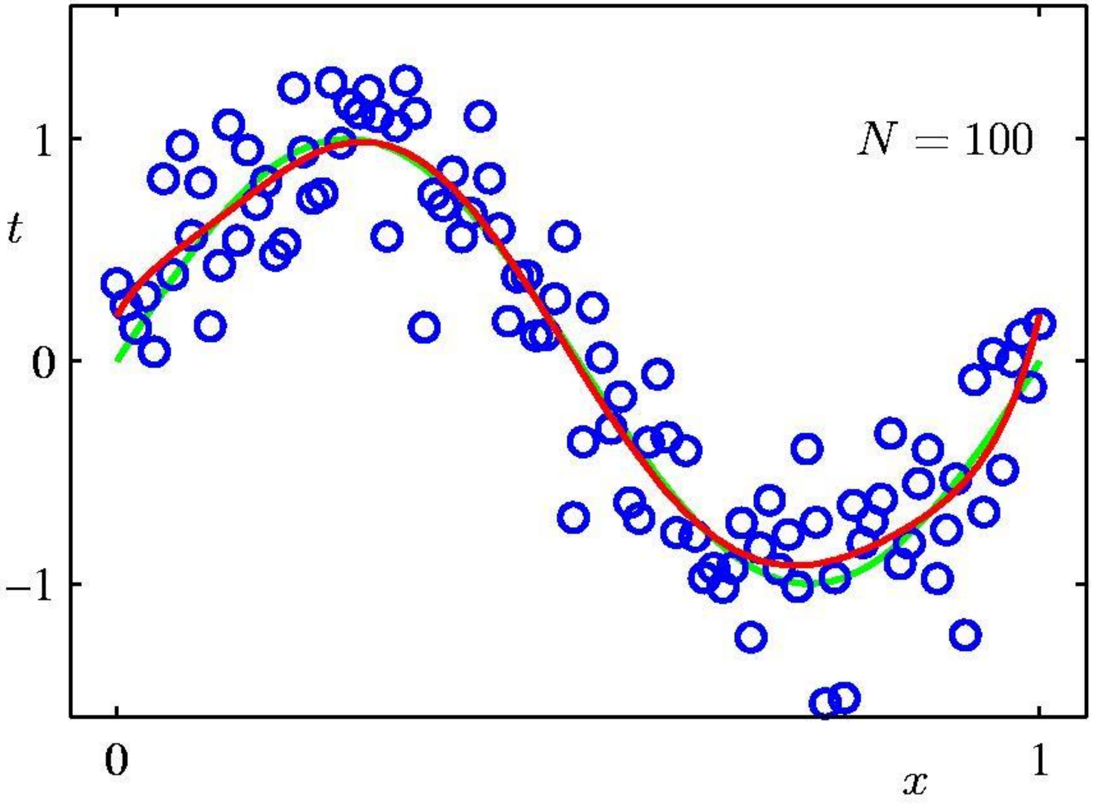
Data Set Size: $N = 15$

9th Order Polynomial



Data Set Size: $N = 100$

9th Order Polynomial

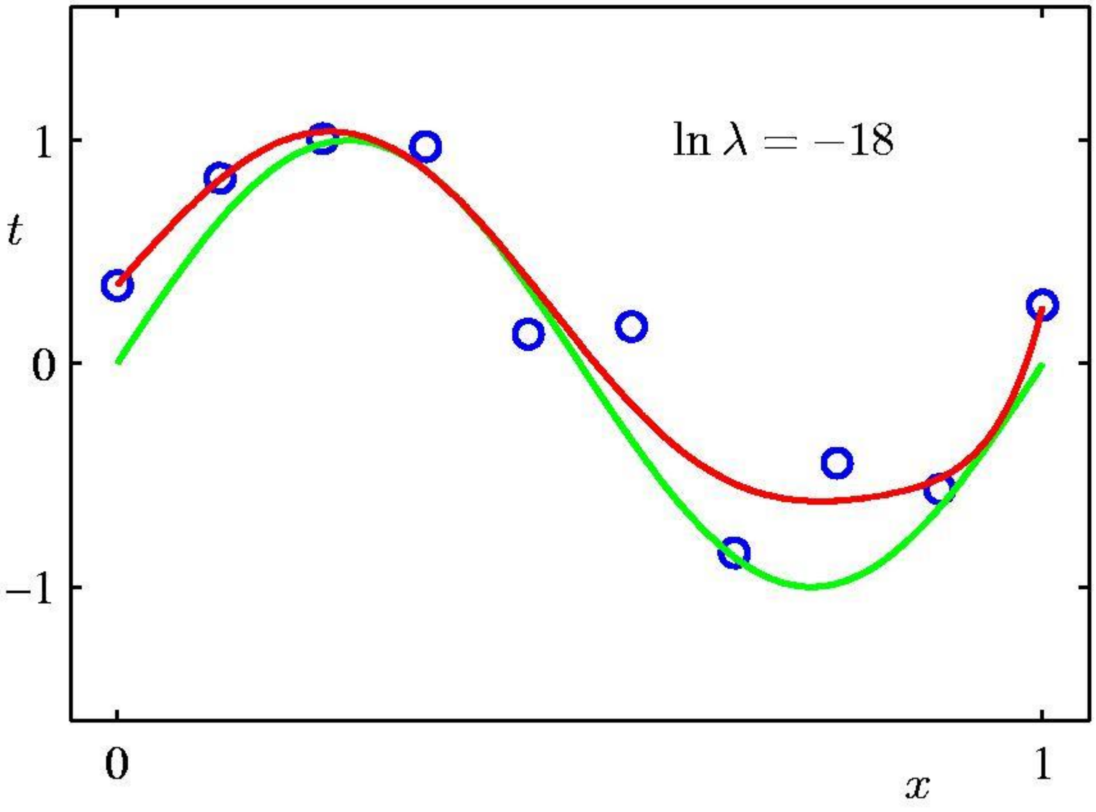


Regularization

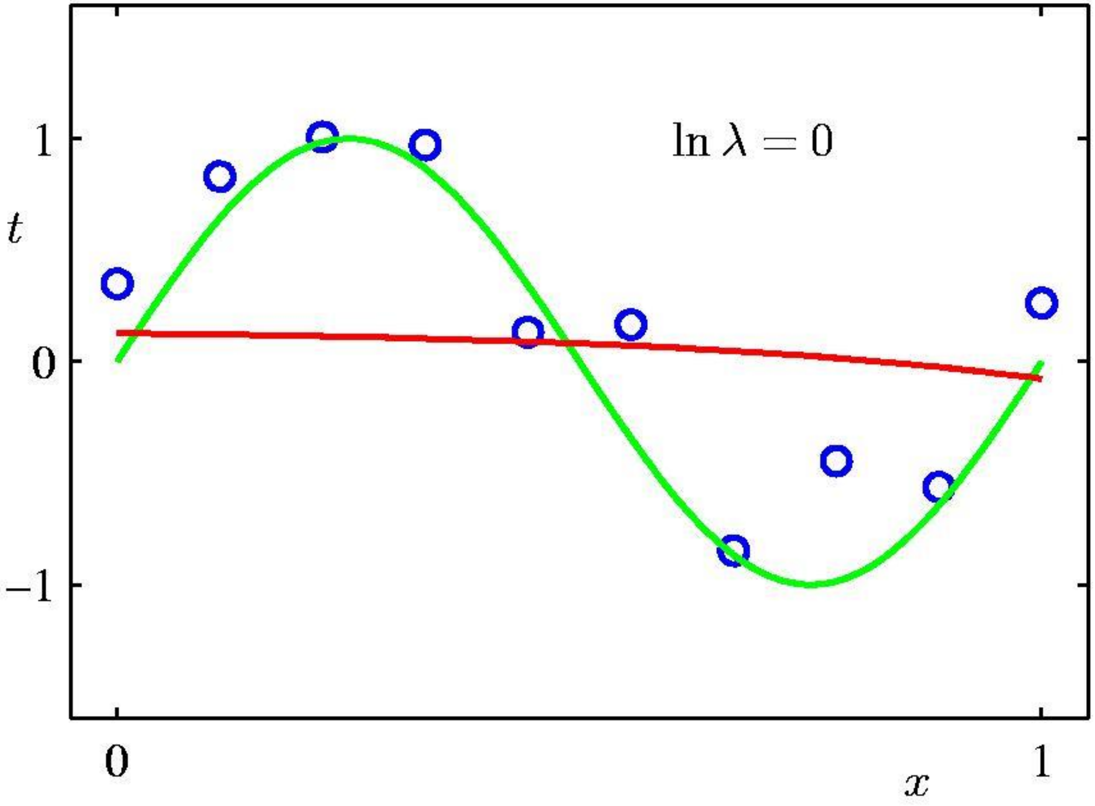
Penalize large coefficient values

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

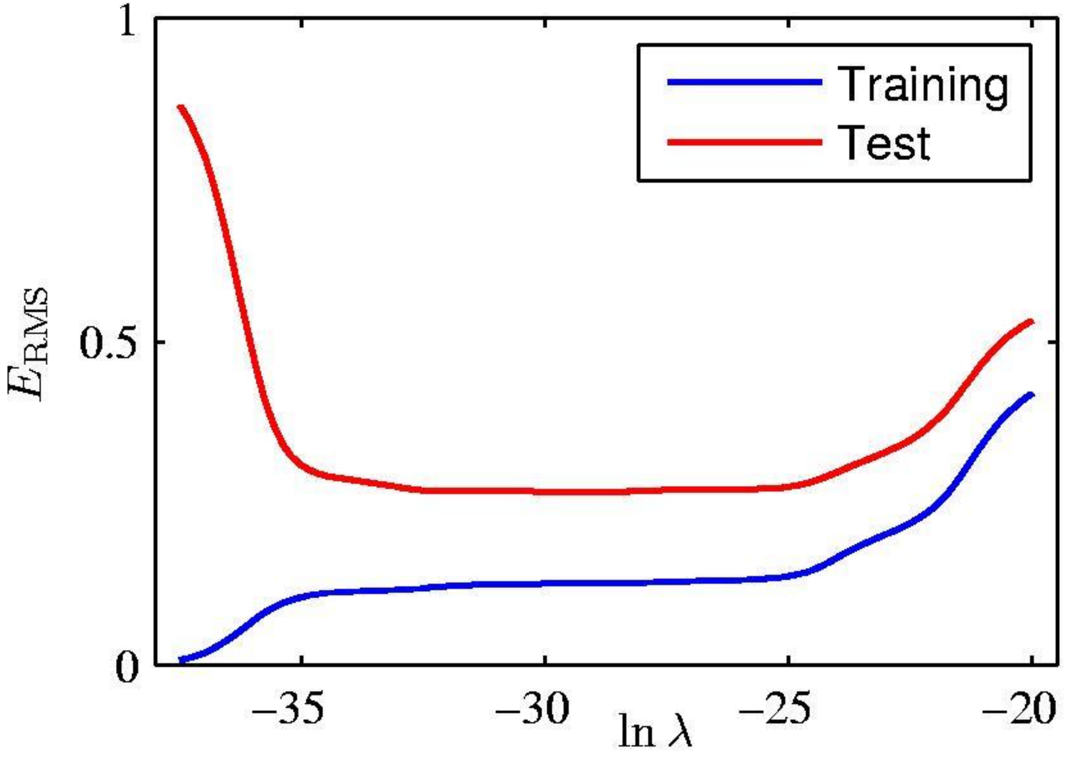
Regularization: $\ln \lambda = -18$



Regularization: $\ln \lambda = 0$



Regularization: E_{RMS} vs. $\ln \lambda$



Polynomial Coefficients

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Binary Classification

Regression vs Classification

y = f(x)

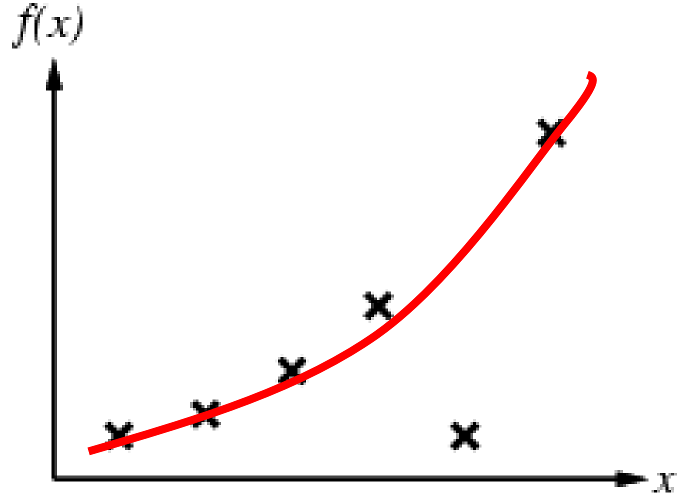
Regression:

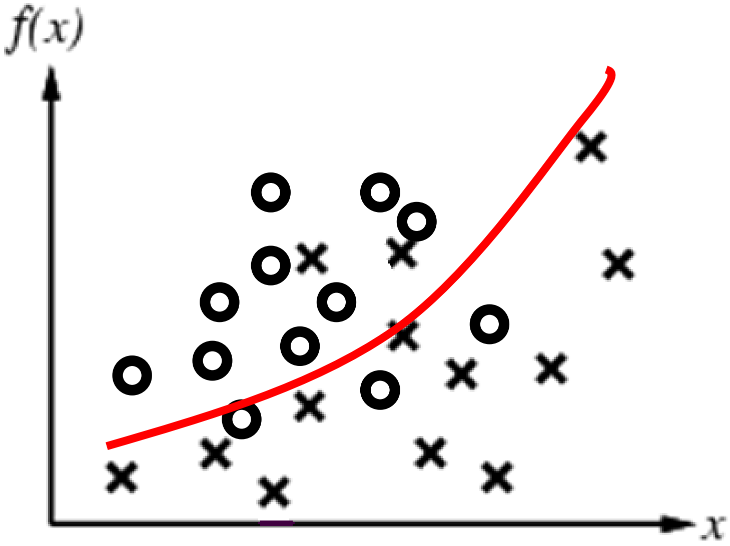
y is continuous

Classification:

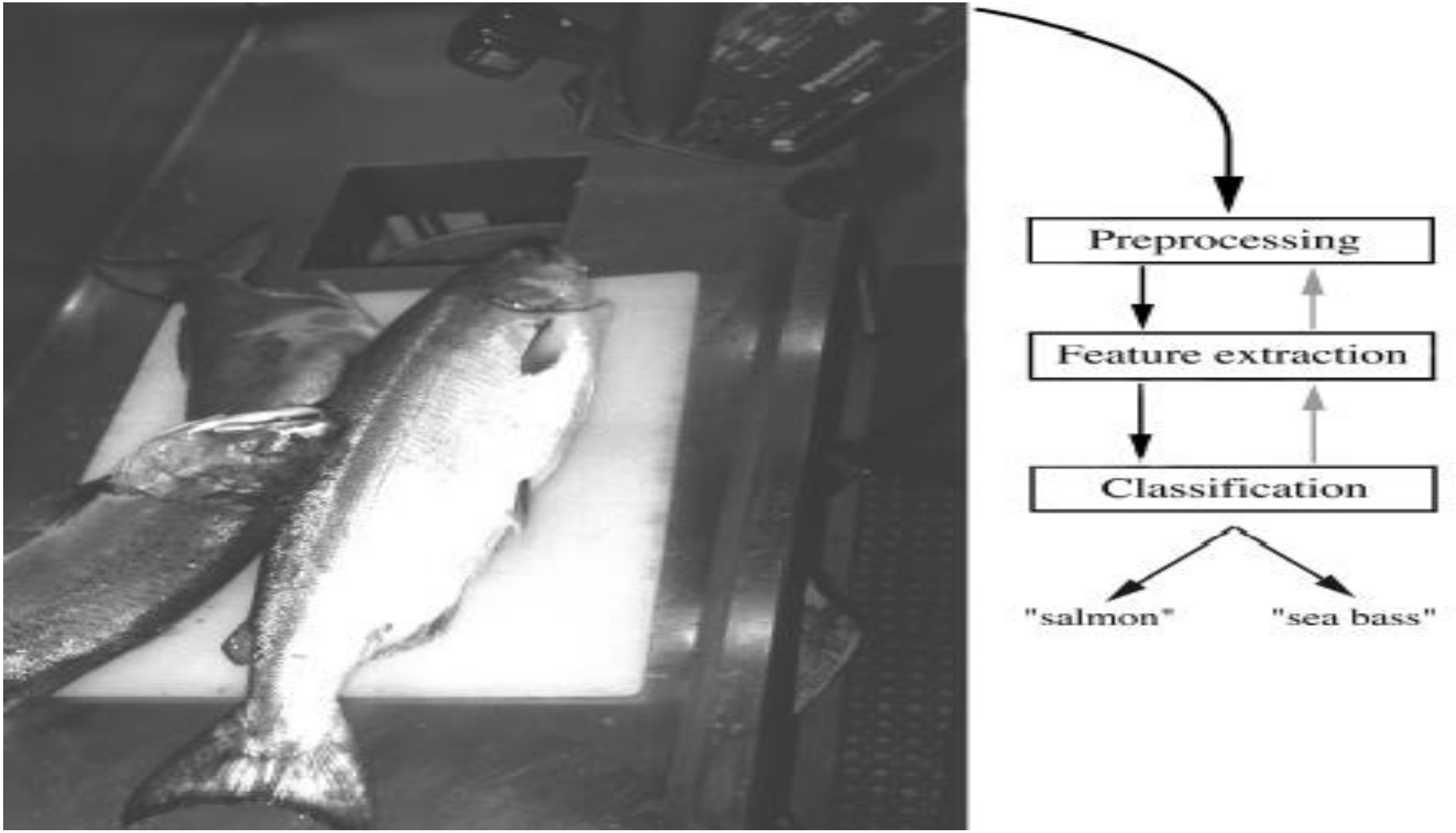
y : discrete values e.g. 0,1,2...
for classes C_0, C_1, C_2...

Binary Classification: two classes
y ∈ {0,1}





Binary Classification



Preprocessing

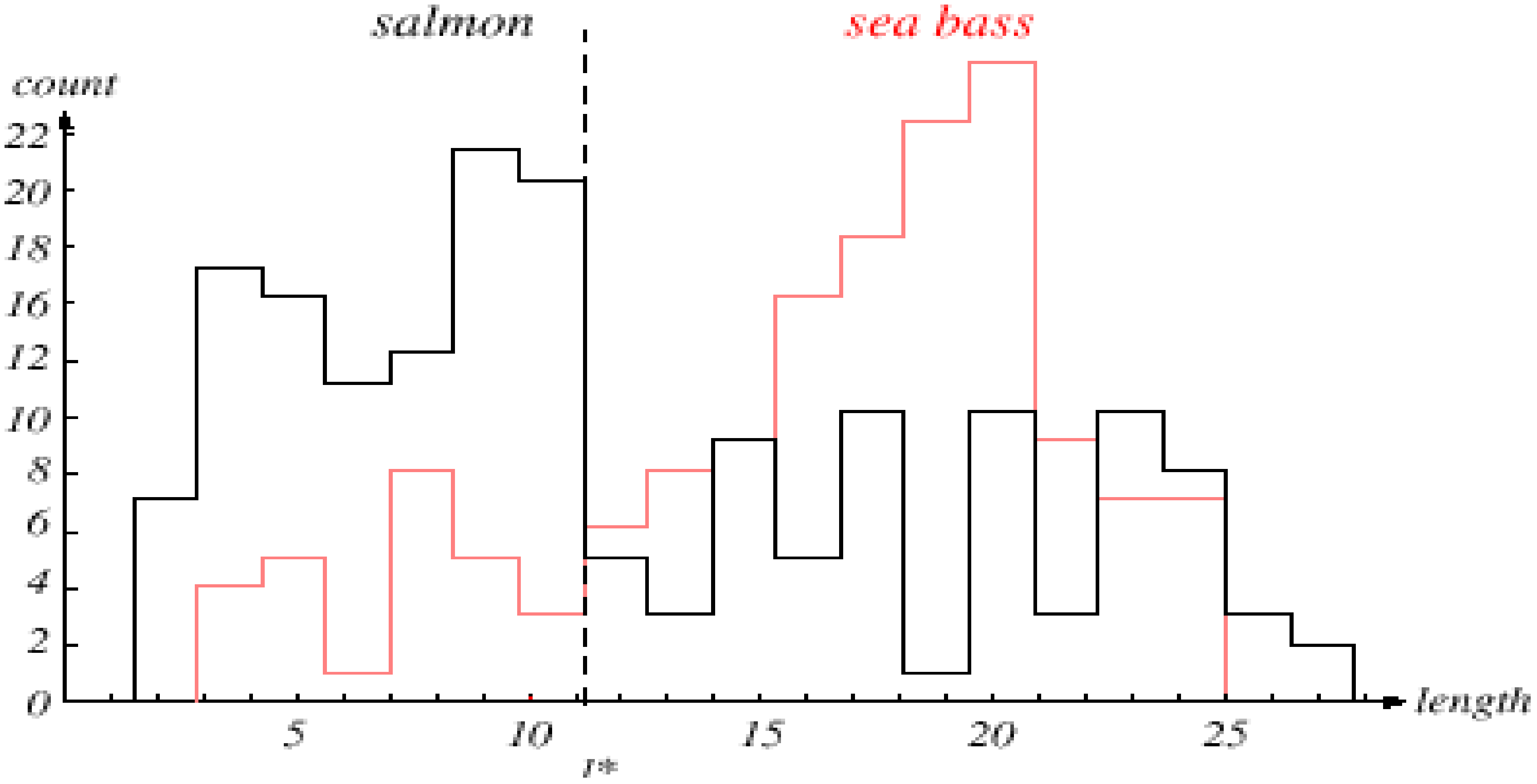
Feature extraction

Classification

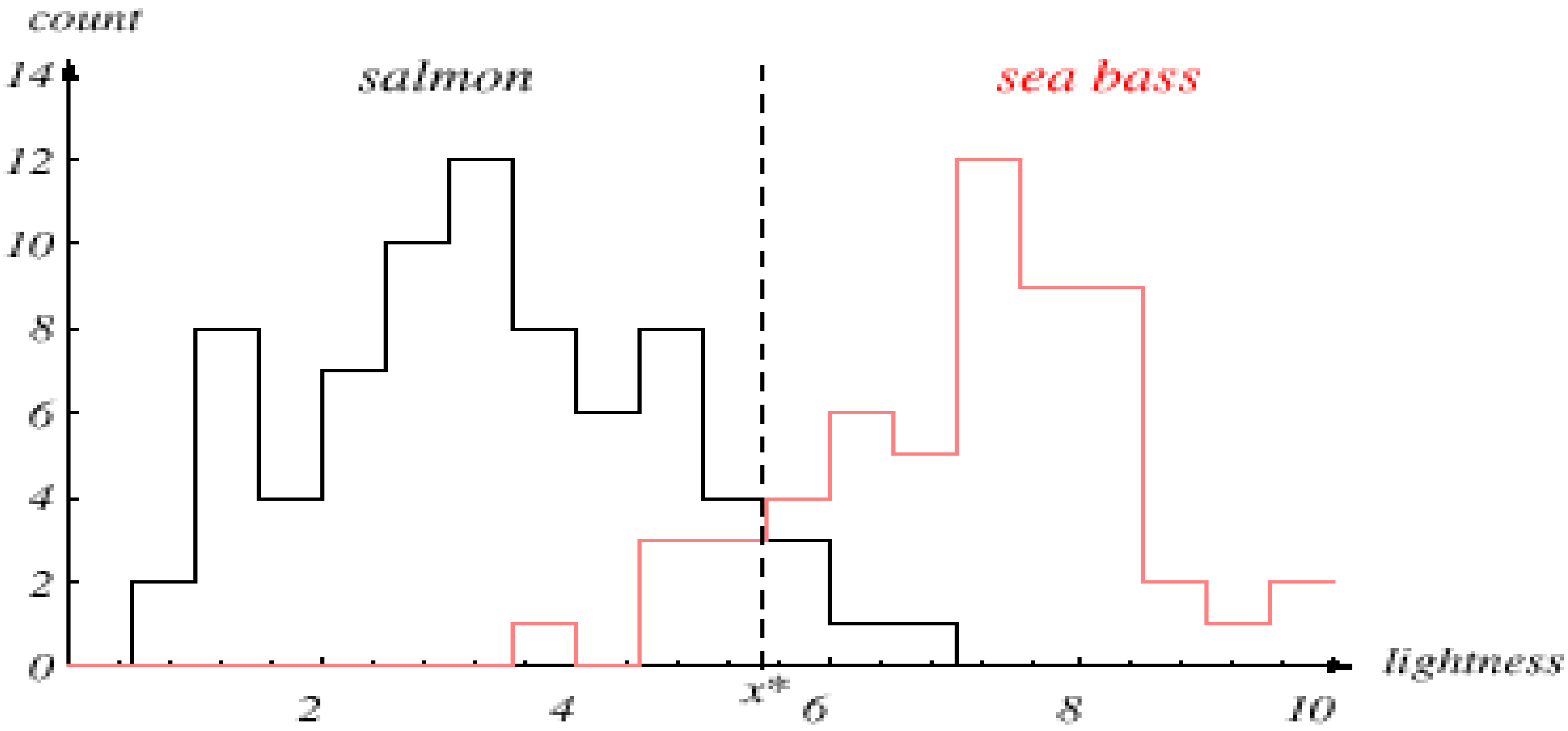
"salmon"

"sea bass"

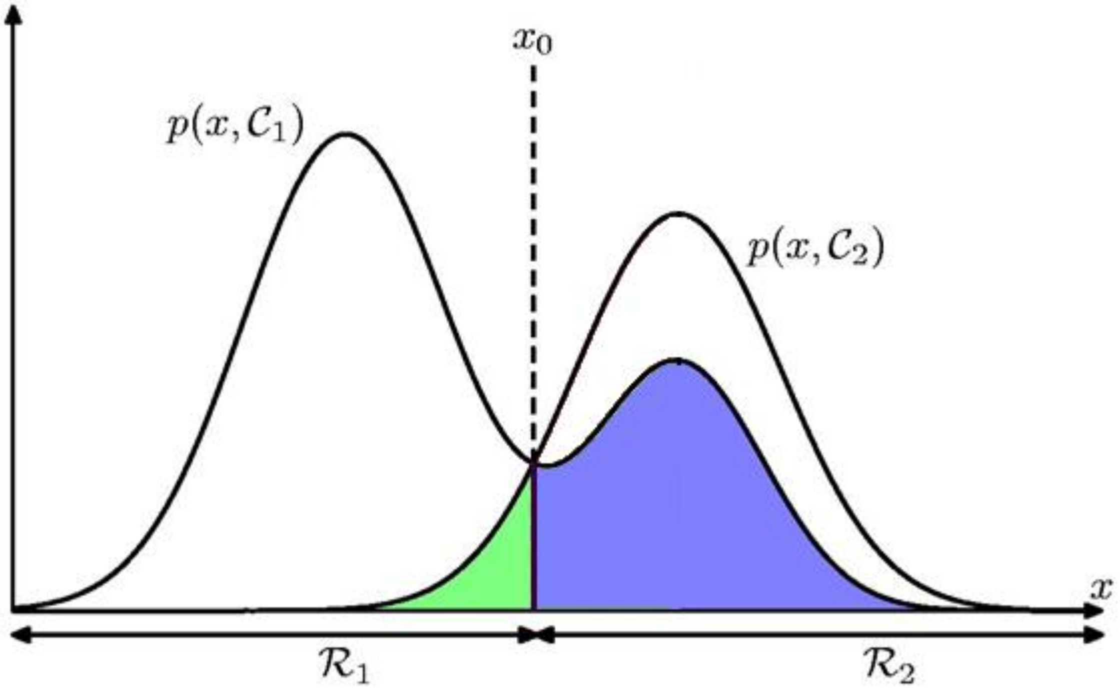
Feature : Length



Feature : Lightness



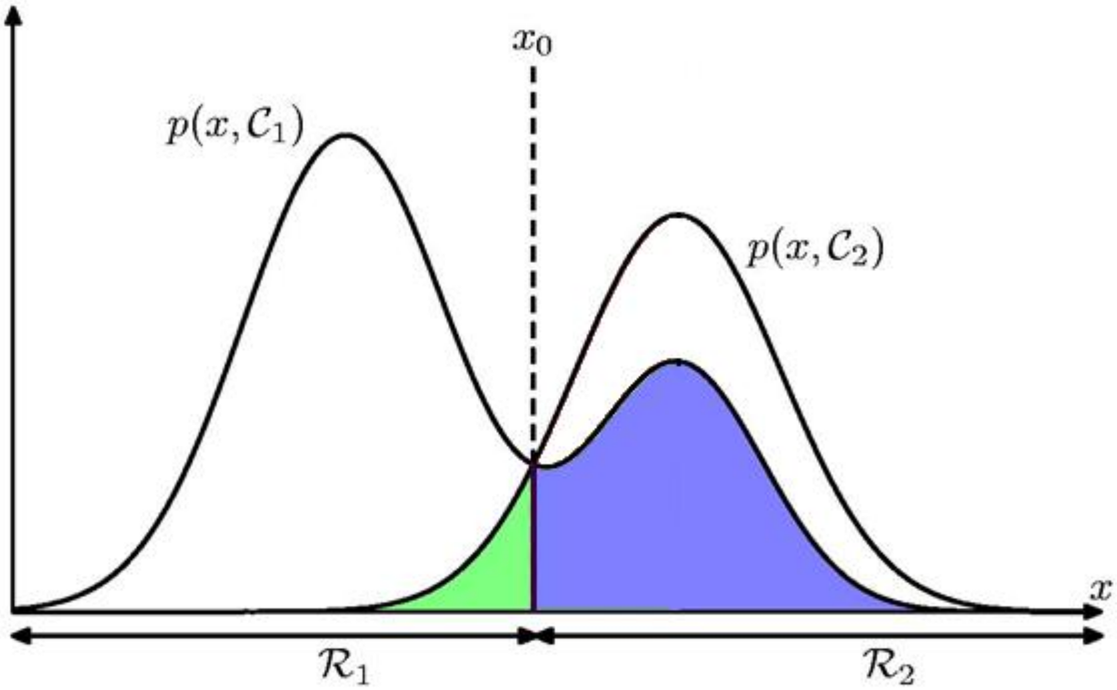
Minimize Misclassification



$$
\begin{aligned}
p(\text{mistake}) &= p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1) \\
&= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2)\,\mathrm{d}\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1)\,\mathrm{d}\mathbf{x}.
\end{aligned}
$$

Precision / Recall

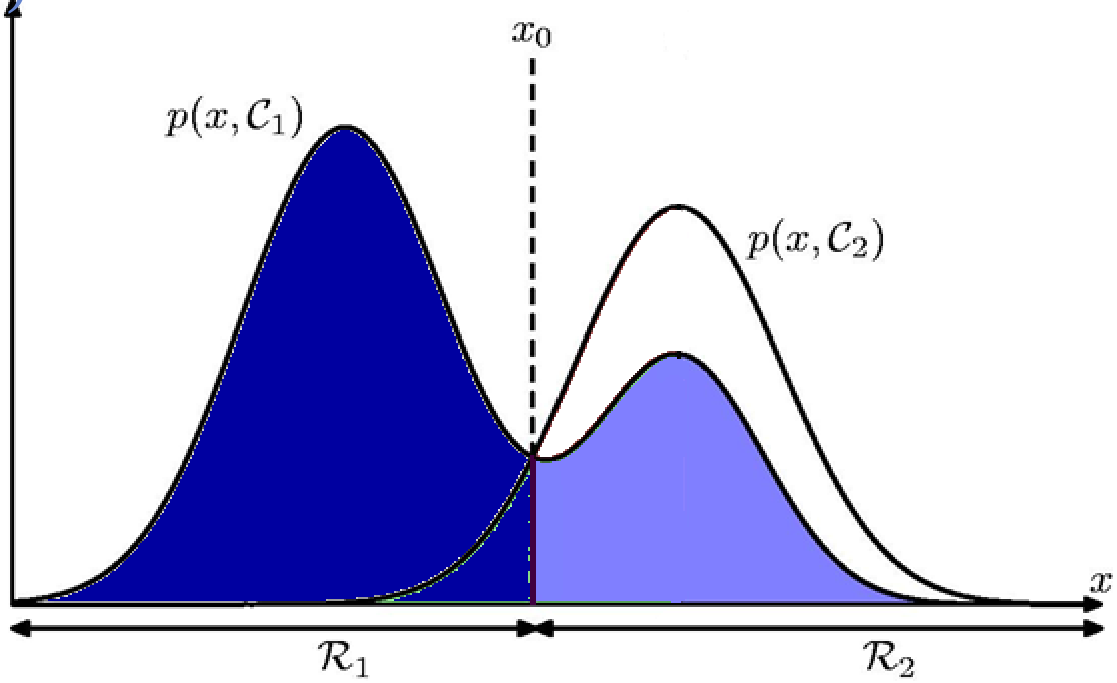
C1 : class of interest



Which is higher: Precision, or Recall?

Precision / Recall

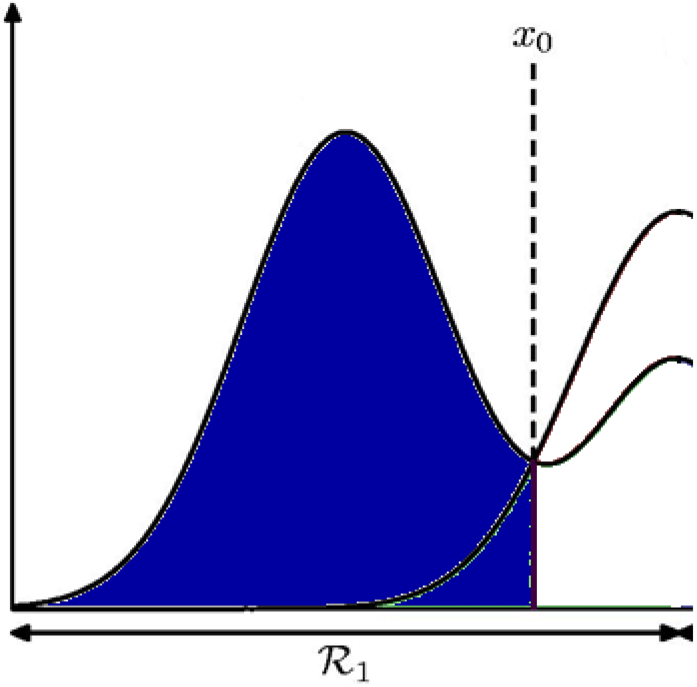
C1 : class of interest (Positives)

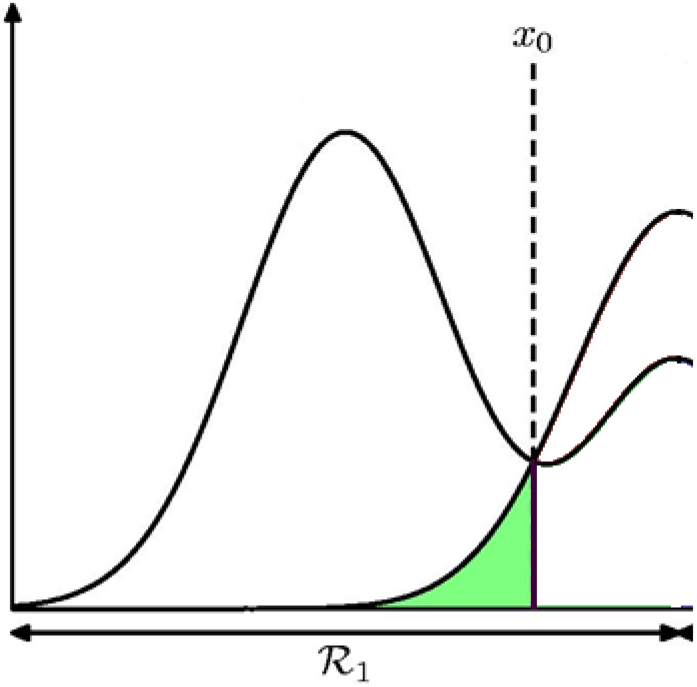


Recall = TP / TP +FP

Precision / Recall

C1 : class of interest





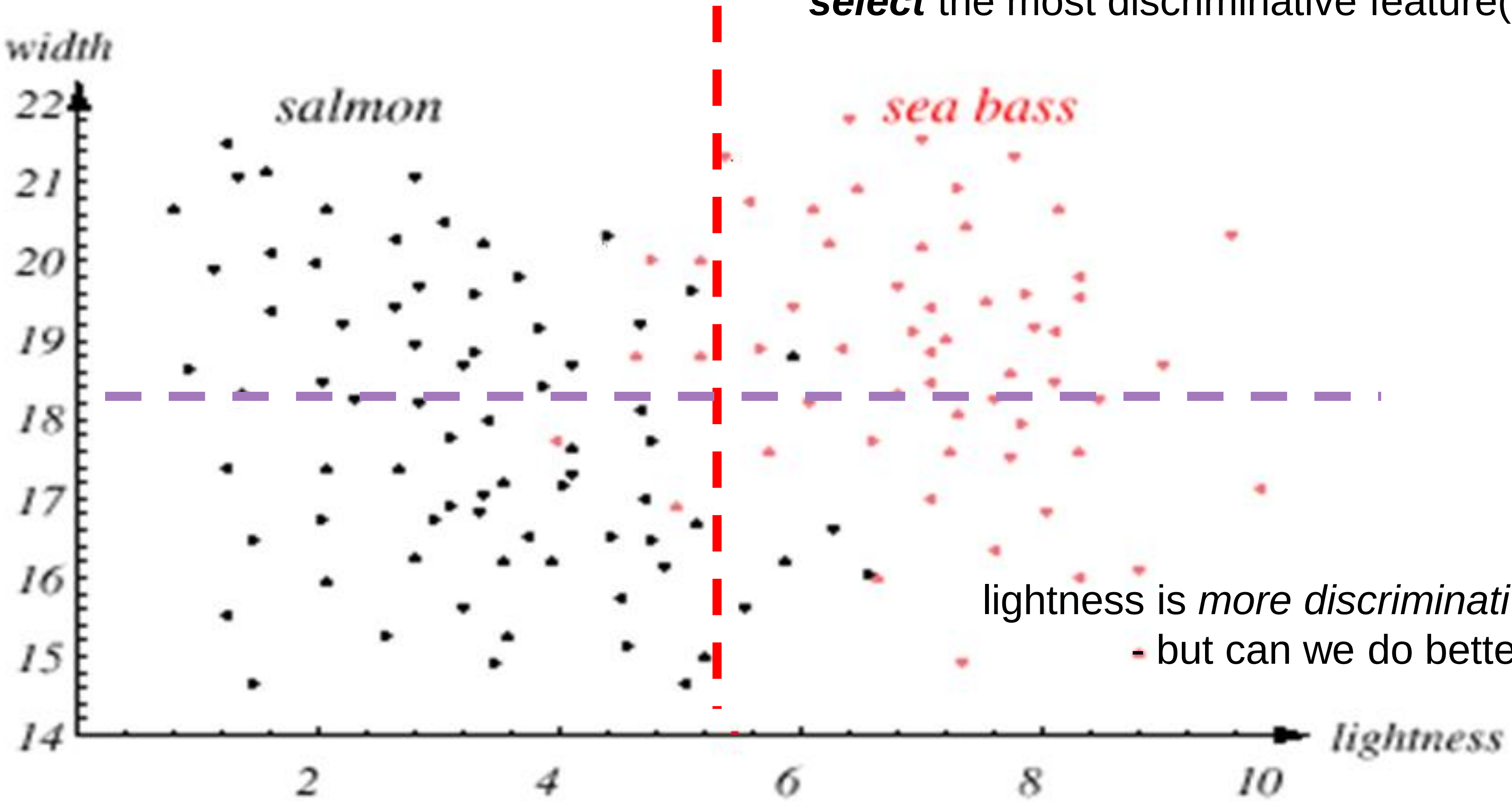
Precision = TP / TP +FN

Decisions - Feature Space

- Feature selection : which feature is maximally discriminative?
  - Axis-oriented decision boundaries in feature space
  - Length – or – Width – or Lightness?

- Feature Discovery: construct g(), defined on the feature space, for better discrimination

Feature Selection: *width / lightness*



select the most discriminative feature(s)

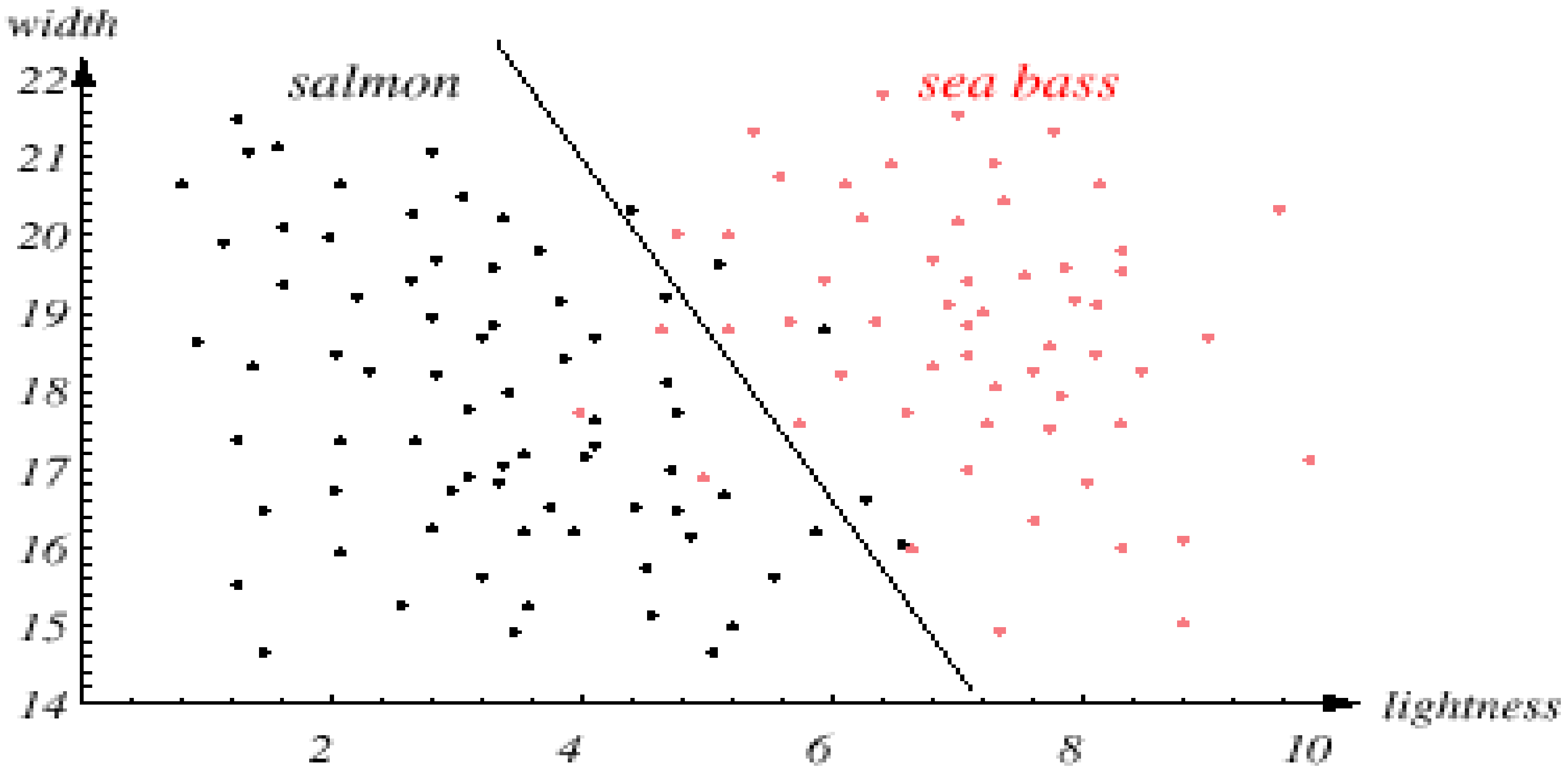
lightness is *more discriminative*
- but can we do better?

Feature Selection

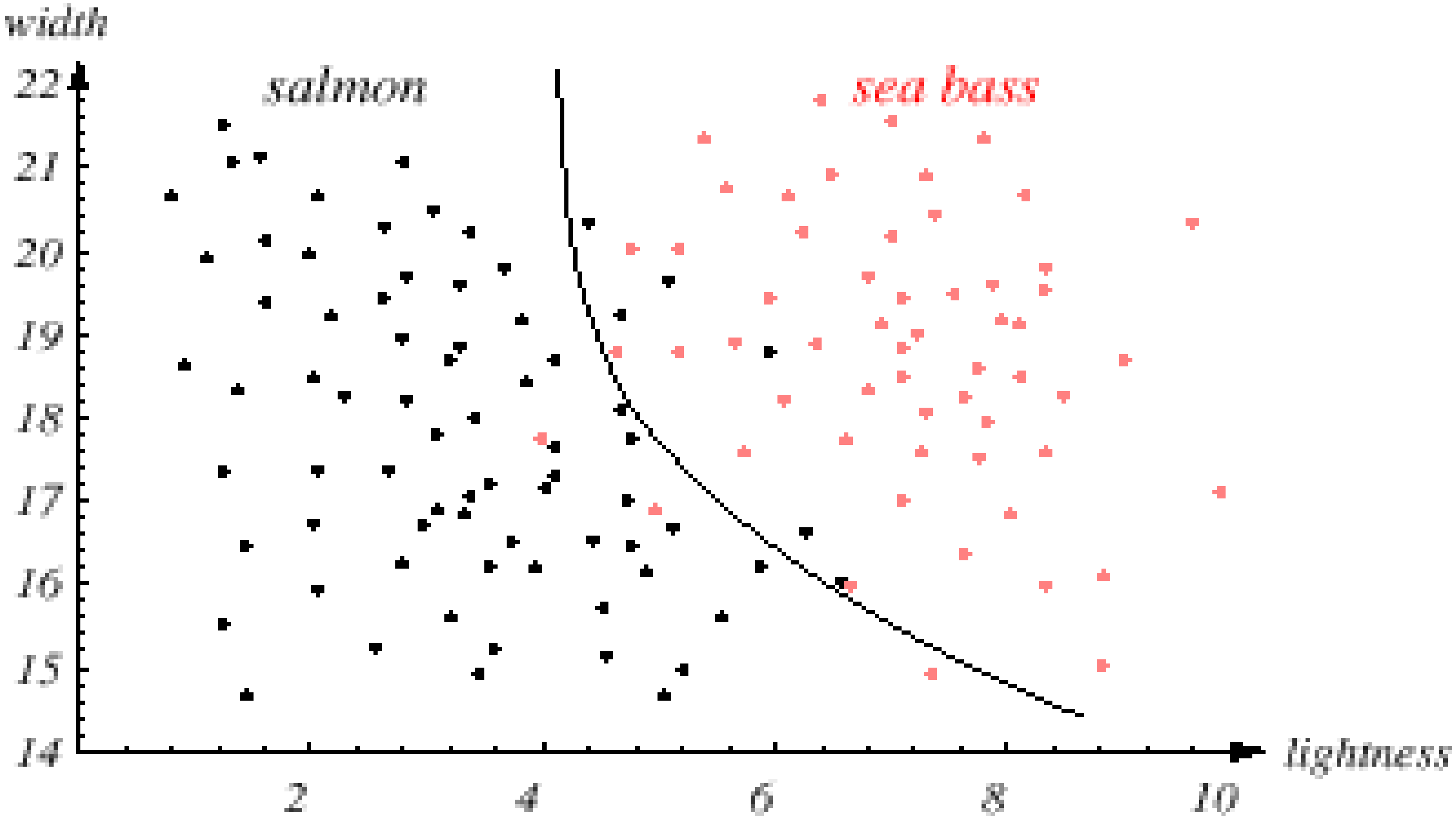
- Feature selection : which feature is maximally discriminative?
  - Axis-oriented decision boundaries in feature space
  - Length – or – Width – or Lightness?

- Feature Discovery: discover discriminative function on feature space : g()
  - combine aspects of length, width, lightness

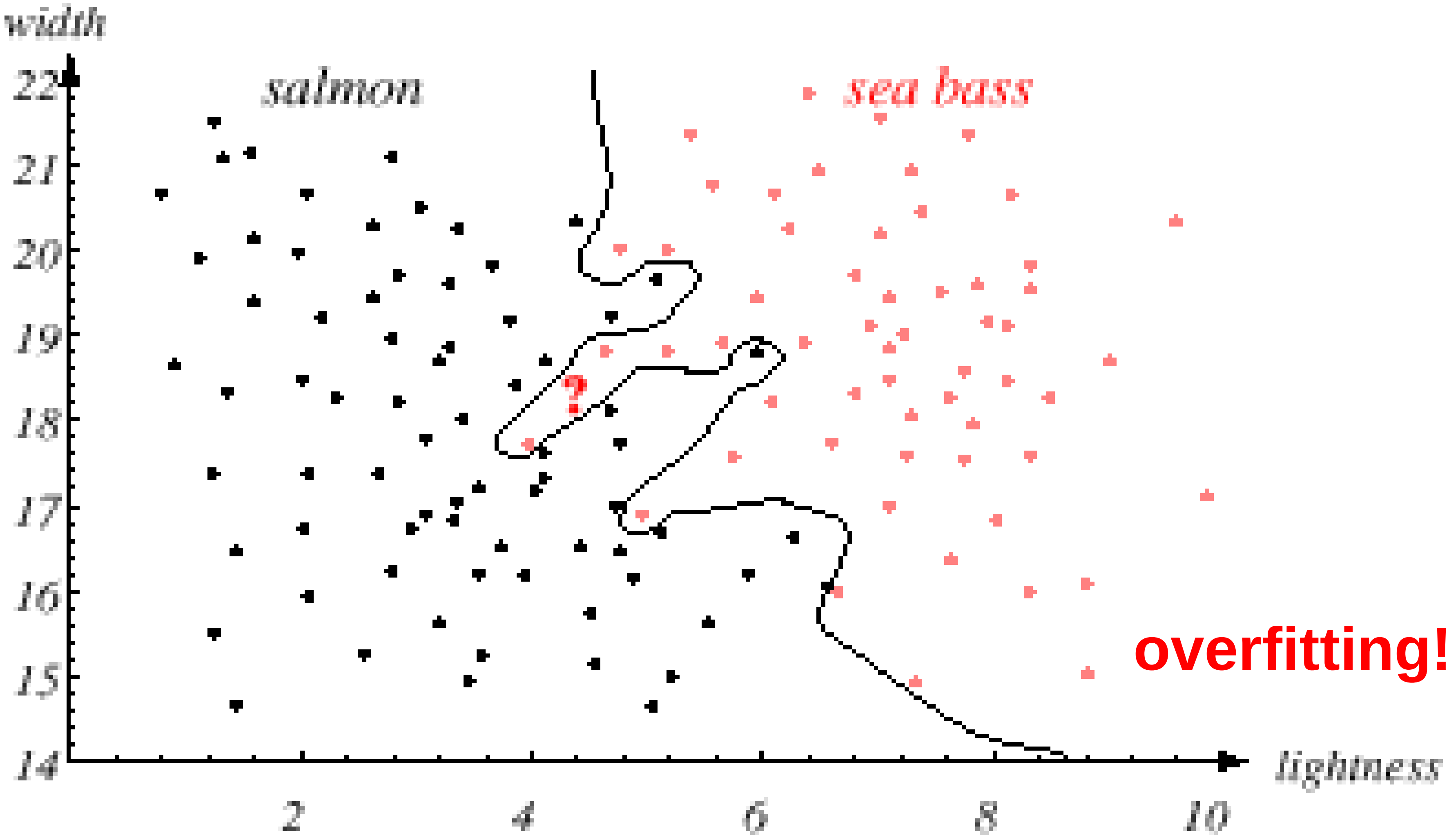
Feature Discovery : Linear



Decision Surface: non-linear



Decision Surface : non-linear



overfitting!

Learning process

- Feature set : representative? complete?

- Sample size : training set vs test set

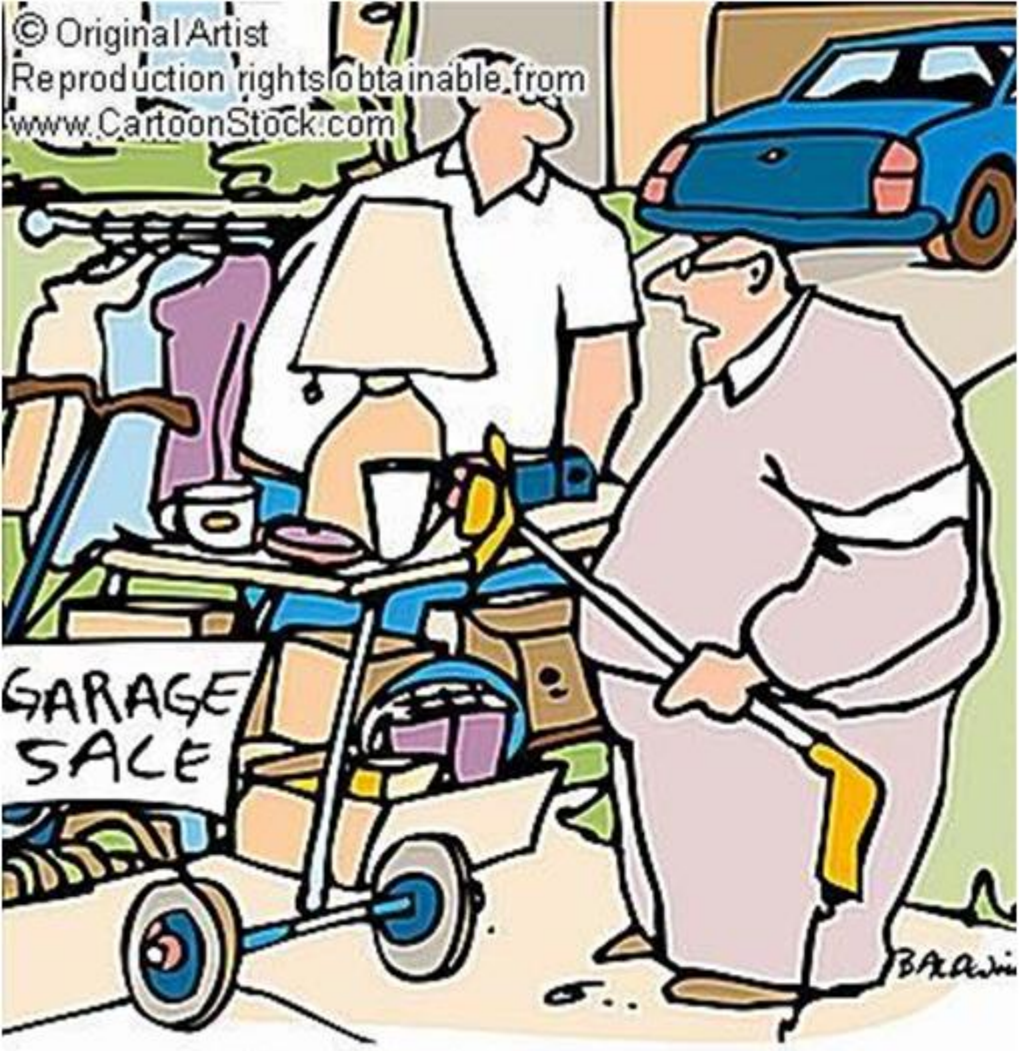
- Model selection:

  - Unseen data → overfitting?
  - Quality vs Complexity
  - Computation vs Performance

Best Feature set?

- Is it possible to describe the variation in the data in terms of a compact set of Features?

- Minimum Description Length



"You spelled garbage wrong."

CS 671 NLP NAIVE BAYES AND SPELLING

amitabha mukerjee

iit kanpur

Reading

- □ Reading:

1. Chapter 6 of Jurafsky & Martin, Speech and Language Processing, “Spelling Correction noisy channel“ (draft 2014 edition) http://web.stanford.edu/~jurafsky/slp3/

2. P. Norvig, How to write a spelling corrector http://norvig.com/spell-correct.html

Spelling Correction

In [2], the authors used curvatures for accurate loacation and tracking of the center of the eye.

OpenCV has cascades for faces whih have been used for detcting faces in live videos.

- course project report 2013

black crows gorge on bright mangoes in still, dustgreen trees

→ ?? "black cows" ?? "black crews" ??

Single-typing errors

- loacation : insertion error
- whih , detcting : deletion
- crows -> crews : substitution
- the -> hte : transposition

Damereau (1964) : 80% of all misspelled words caused by single-error of these four types

Which errors have a higher “edit-distance”?

Causes of Spelling Errors

- □ Keyboard Based
  - 83% novice and 51% overall were keyboard related errors
  - Immediately adjacent keys in the same row of the keyboard (50% of the novice substitutions, 31% of all substitutions)
- □ Cognitive : may be more than 1-error; more likely to be real words
  - Phonetic : separate → separate
  - Homonym : piece → peace ; there → their;

Steps in spelling correction

Non-word errors:

- Detection of non-words (e.g. hte, dtection)
- Isolated word error correction

[naive bayesian; edit distances]

Actual word (real-word) errors:

- Context dependent error detection and correction (e.g. “three are four types of errors”)

[can use language models e.g. n-grams]

Nonword and Word errors

loacation, detecting → non-words

crews / crows → word error

Non-word error:

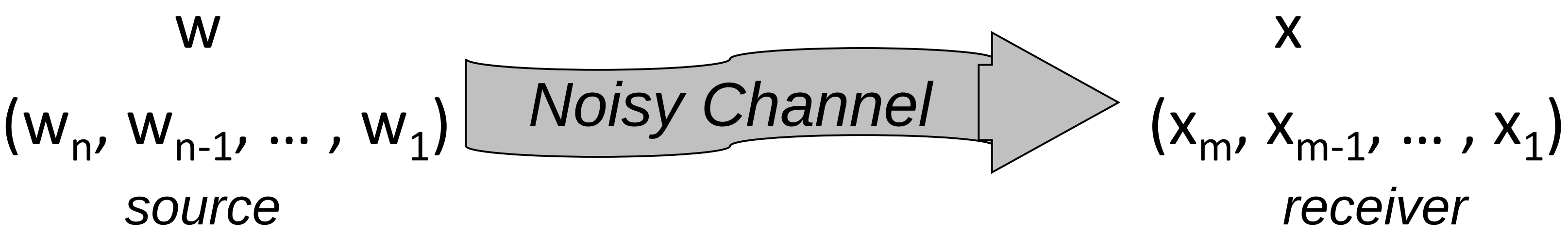
For alphabet Σ, and dictionary D with strings in Σ^*

given a string $s \in \Sigma^*$, where $s \notin D$,

find $w \in D$ that is most likely to have been input as s.

Word error: drop $s \notin D$

Probabilistic Spell Checker



w

$(w_n, w_{n-1}, \ldots, w_1)$

source

Noisy Channel

x

$(x_m, x_{m-1}, \ldots, x_1)$

receiver

Given t, find most probable w :

Find that ŵ for which *P(w|t)* is maximum,

$$\hat{w} = \underset{w \,\epsilon\, V}{argmax}\, P(w|x)$$

best guess

Vocabulary

intended word

mis-spelled word

Probabilistic Spell Checker

- Q. How to compute $P(w|t)$?

- *Many times, it is easier to compute* $P(t|w)$

Bayesian Classification

- Given an observation x, determine which class w it belongs to
- Spelling Correction:
  - Observation: String of characters
  - Classification: Word intended
- Speech Recognition:
  - Observation: String of phones
  - Classification: Word that was said

PROBABILITY THEORY

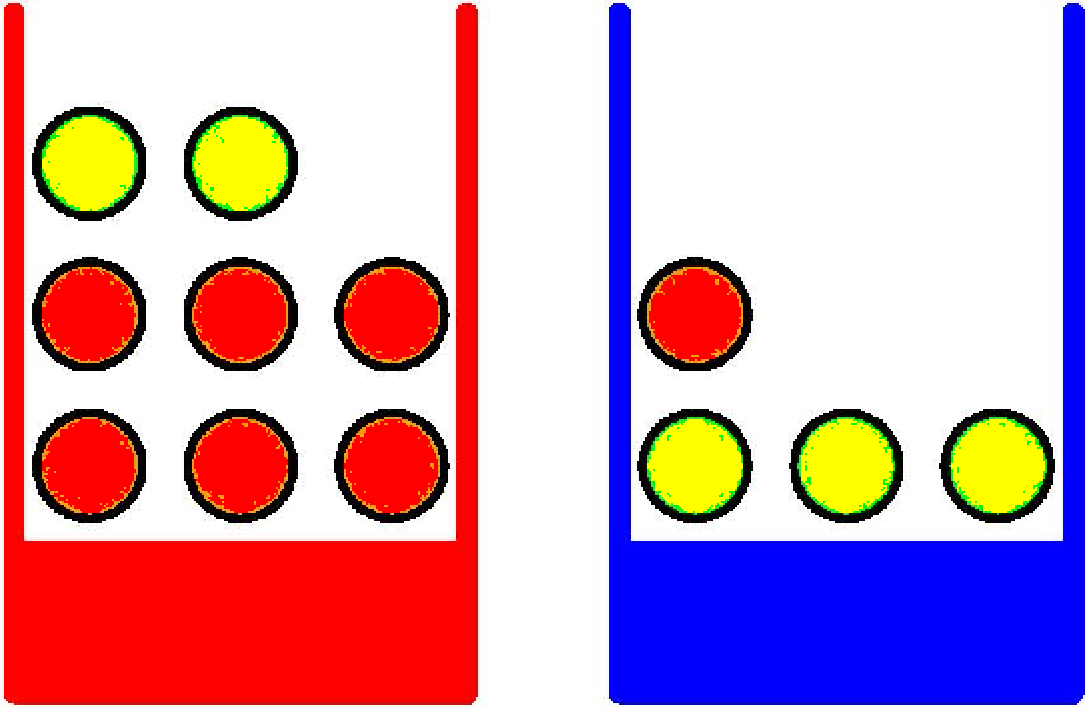
Example

AIDS occurs in 0.05% of population. A test is 99% effective in detecting the disease, but 5% of the cases test positive in absence of *AIDS.*

If you are tested +ve, what is the probability you have the disease?

Probability theory

Apples and Oranges



Sample Space

Sample ω = Pick two fruits,

e.g. Apple, then Orange

Sample Space Ω = {(A,A), (A,O),

(O,A),(O,O)}

= all possible worlds

Event e = set of possible worlds, $e \subseteq \Omega$

- e.g. second one picked is an apple

Learning = discovering regularities

- **Regularity** : repeated experiments:
  outcome not be fully predictable

- **Probability** p(e) : "the fraction of possible worlds in which e is true” i.e. outcome is event e

- **Frequentist** view : p(e) = limit as N → ∞

- **Belief** view: in wager : equivalent odds
  (1-p):p that outcome is in e, or vice versa

Why probability theory?

different methodologies attempted for uncertainty:

- Fuzzy logic
- Multi-valued logic
- Non-monotonic reasoning

But **unique property** of probability theory:

If you gamble using probabilities you have the best chance in a wager. [de Finetti 1931]

=> if opponent uses some other system, he's more likely to lose

Ramsay-diFinetti theorem (1931)

If agent X’s degrees of belief are **rational**, then X ’s degrees of belief function defined by **fair betting** rates is (formally) a probability function

Fair betting rates: opponent decides which side one bets on

Proof: fair odds result in a function pr () that satisifies the Kolmogrov axioms:

Normality : pr(S) >=0

Certainty : pr(T)=1

Additivity : pr (S1 v S2 v..)= Σ(Si)

Kolmogrovian model

Probability space Ω = set of all outcomes (events)

Event A may include multiple outcomes – e.g. several coin-tosses.

F : a σ-field on Ω : closed under countable union, and under complement, maximal element Ω, emptySet= impossible event

In practice, *F* = all possible subsets = powerset of Ω

(alternatives to kolmogrovian axiomatization exist)

Axioms of Probability

A probability measure p : F → [0,1], s.t.

- p is **non-negative** : $p(e) \geq 0$

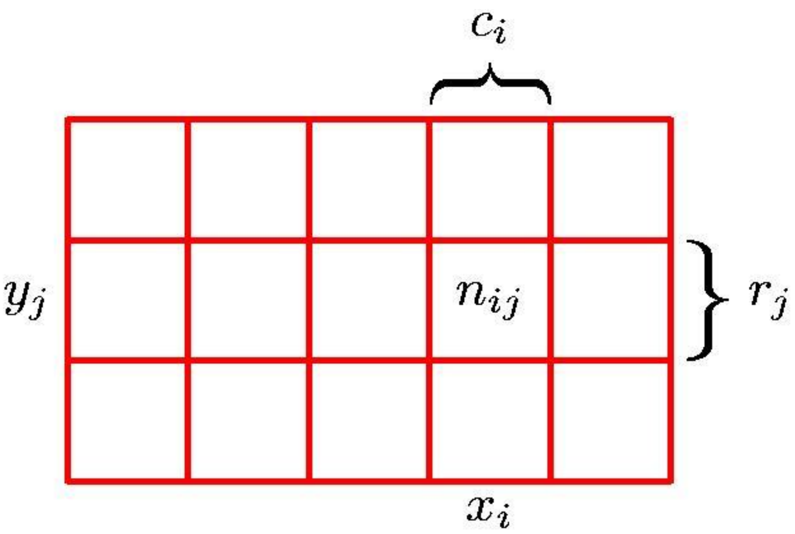
- **unit sum** $p(\Omega) = 1$

  i.e. no outcomes outside sample space

- **additive** : if e1, e2 are disjoint events (no common outcome):

  $p(e1) + p(e2) = p(e1 \cup e2)$

Joint vs. conditional probability



Marginal Probability

$$p(X = x_i) = \frac{c_i}{N}.$$

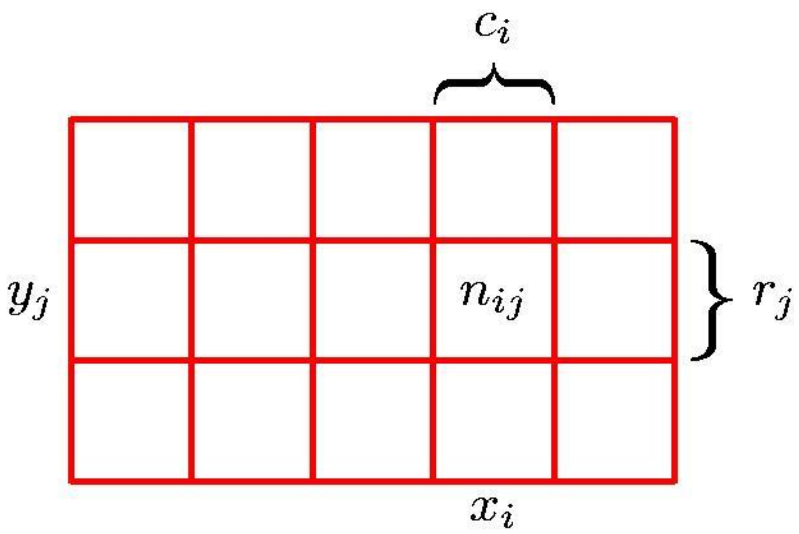
Joint Probability

$$p(X = x_i, Y = y_j) = \frac{n_{ij}}{N}$$

Conditional Probability

$$p(Y = y_j | X = x_i) = \frac{n_{ij}}{c_i}$$

Probability Theory



Sum Rule

$$\begin{aligned} p(X = x_i) = \frac{c_i}{N} &= \frac{1}{N}\sum_{j=1}^{L} n_{ij} \\ &= \sum_{j=1}^{L} p(X = x_i, Y = y_j) \end{aligned}$$

Product Rule

$$\begin{aligned} p(X = x_i, Y = y_j) &= \frac{n_{ij}}{N} = \frac{n_{ij}}{c_i} \cdot \frac{c_i}{N} \\ &= p(Y = y_j | X = x_i) p(X = x_i) \end{aligned}$$

Rules of Probability

Sum Rule $$p(X) = \sum_Y p(X, Y)$$

Product Rule $$p(X, Y) = p(Y|X)p(X)$$

Example

parasitic Gap, a rare syntactic construction occurs on average once in 100,000 sentences.

pattern matcher : find sentences S w parasitic gaps.

if S has parasitic gap (G), → says (T) with prob 0.95.

if S has no gap (~G) wrongly says (T) w prob 0.005.

On a corpus of 100000 Sentences, How many are expected to be detected with G?

$P(G) = 10^{-5}$. $P(T|G) = 0.95$ $P(T|{\sim}G) = 0.005 = 5.10^{-3}$

truly G = 0.95 ; falsely detected as G = 500

Probabilistic Spell Checker

- Q. How to compute $P(w|t)$?

- *Many times, it is easier to compute $P(t|w)$*

- Related by product rule:

$$\begin{aligned} p(X,Y) &= p(Y|X)\,p(X) \\ &= p(X|Y)\,p(Y) \end{aligned}$$

Bayes' Theorem

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}$$

$$p(X) = \sum_{Y} p(X|Y)p(Y)$$

posterior $\propto$ likelihood × prior

Bayes' Theorem

Thomas Bayes (c.1750):

how can we infer causes from effects?

can one learn the probability of a future event from frequency of occurrance in the past?

as new evidence comes in → probabilistic knowledge improves.

→ basis for human expertise?

Initial estimate (*prior* belief *P(h)*, not well formulated)

+ new evidence (support)

+ compute likelihood *P (data| h)*

→ improved *posterior: P (h| data)*

Example

parasitic Gap, a rare syntactic construction occurs on average once in 100,000 sentences.

pattern matcher : find sentences S w parasitic gaps.

if S has parasitic gap (G), → says (T) with prob 0.95.

if S has no gap (~G) wrongly says (T) w prob 0.005.

If the test is positive (T) for a sentence, what is the probability that there is a parasitic gap?

$P(G) = 10^{-5}$. $P(T|G) = 0.95$ $P(T|\sim G) = 0.005 = 5.10^{-3}$

truly G = 0.95 ; falsely detected as G = 500

Example

$P(G) = 10^{-5}$. $P(T|G) = 0.95$ $P(T|\sim G) = 0.0005 = 5.10^{-4}$

$P(G|T) = P(T|G) * P(G) / P(T)$

$P(T) = P(T,G) + P(T,\sim G)$) [Sum Rule]

$= P(T|G) * P(G) + P(T|\sim G) * P(\sim G)$ [Product Rule]

P(G|T) = 0.95 * 10^-5 / [.95* 10**(-5) + 5.10^-3 . (1 - 10^-5)]
= 9.5e-4 / (9.5e-4 + 5 * 0.99999) [div by 10^-3]
= 0.0095 / (0.0095 + 4.9995) = 0.0095 / 5.00945
= 0.0019

or about 1/500

Bernoulli Process

- □ Two Outcomes – e.g. toss a coin three times:

  HHH, HHT, HTH, HTT, THH, THT, TTH, TTT

- □ Probability of k Heads:

k	0	1	2	3
P(k)	1/8	3/8	3/8	1/8

Probability of success: p, failure q, then

$$P(k) = \binom{n}{k} p^k q^{n-k}$$

Permutations

$$\frac{N!}{n_1!n_2!n_3!.....n_k!} \overset{def.}{=} \binom{N}{n_1, n_2, n_3, ..., n_k}$$

Multinomial Coefficient

K = 2 → Binomial coefficient

PERMUTATIONS

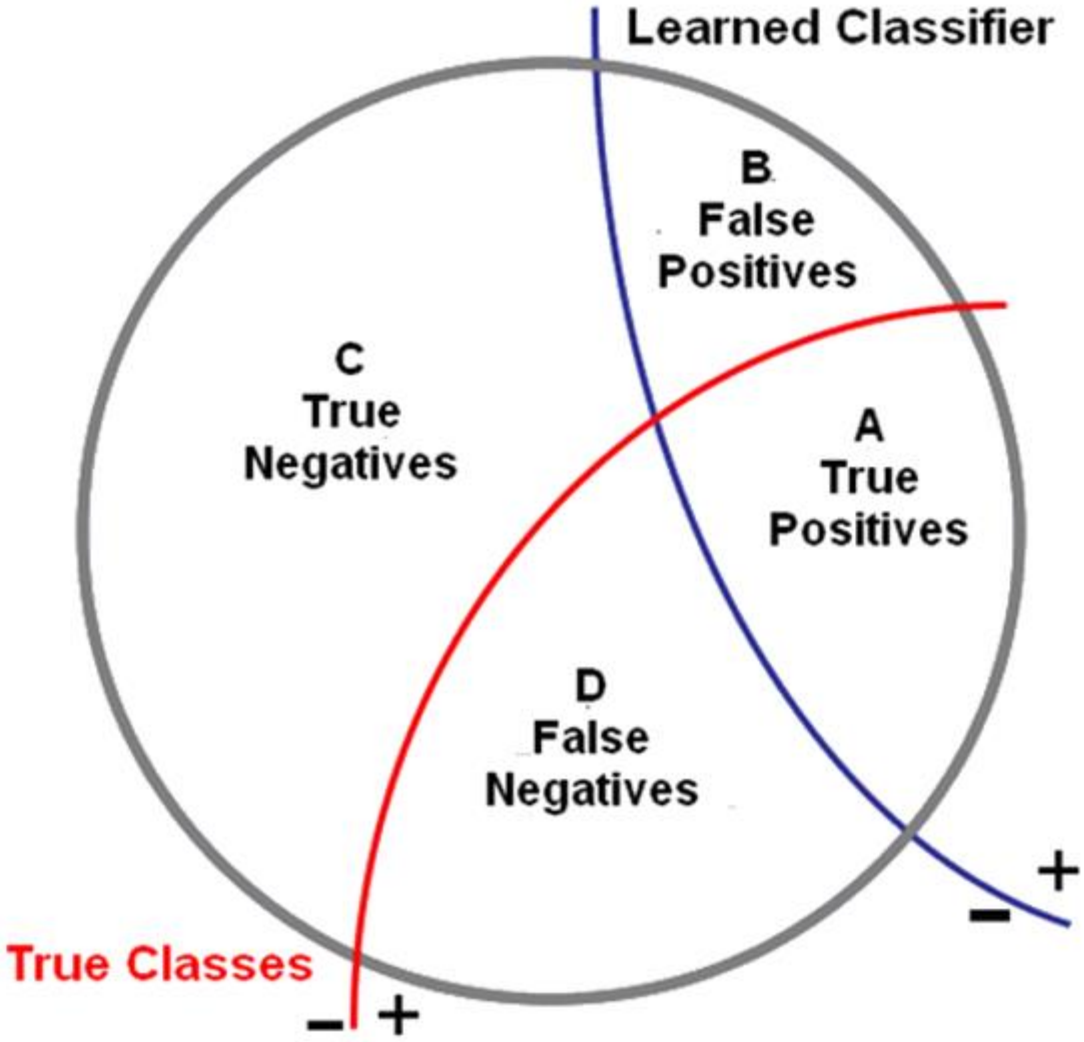
Precision vs Recall

Precision:

A / Retrieved Positives

Recall:

A / Actual Positives



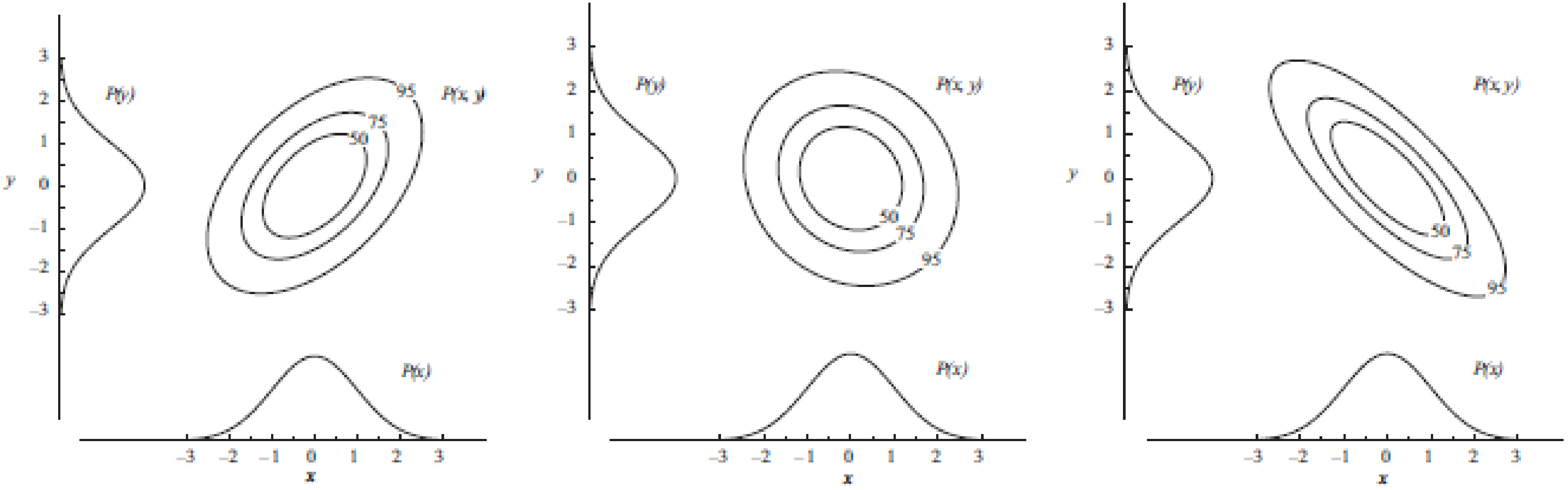
Example

What is the recall of the test for parasitic gap?

What is its precision?

F-Score

Features may be high-dimensional



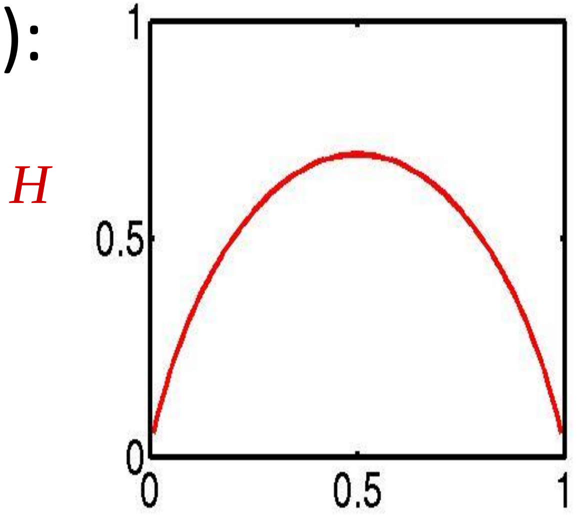
joint distribution P(x,y) varies considerably though marginals P(x), P(y) are identical

estimating the joint distribution requires much larger sample: *O(n*^{k}*)* vs *nk*

Entropy

- Entropy: the uncertainty of a distribution.
- Quantifying uncertainty ("surprisal"):
  - Event x
  - Probability p_x
  - Surprisal $\log(1/p_x)$
- Entropy: expected surprise (over p):

$$\mathrm{H}(p) = E_p\left[\log_2 \frac{1}{p_x}\right] = -\sum_x p_x \log_2 p_x$$



A coin-flip is most uncertain for a fair coin.

NON-WORD SPELL CHECKER

Spelling error as classification

- □ Each word *w* is a class, related to many instances of the observed forms *x*

- □ Assign w given x :

$$\hat{w} = \underset{w \in V}{\operatorname{argmax}}\, P(w \mid x)$$

Noisy Channel : Bayesian Modeling

- □ Observation x of a misspelled word
- □ Find correct word w

$$\hat{w} = \underset{w \in V}{\operatorname{argmax}} P(w \mid x)$$

$$= \underset{w \in V}{\operatorname{argmax}} \frac{P(x \mid w)P(w)}{P(x)}$$

$$= \underset{w \in V}{\operatorname{argmax}} P(x \mid w)P(w)$$

Non-word spelling error example

acress

Confusion Set

Confusion set of word w:

All typed forms t obtainable by a single application of insertion, deletion, substitution or transposition

Confusion set for acress

Error	Candidate Correction	Correct Letter	Error Letter	Type
acress	actress	t	–	deletion
acress	cress	–	a	insertion
acress	caress	ca	ac	transposition
acress	access	c	r	substitution
acress	across	o	e	substitution
acress	acres	–	s	insertion
acress	acres	–	s	insertion

Kernighan et al 90

Confusion set of word w (one edit operation away from w):

- All typed forms t obtainable by a single application of insertion, deletion, substitution or transposition

- Different editing operations have unequal weights
- Insertion and deletion probabilities : conditioned on letter immediately on the left – bigram model.
- Compute probabilities based on training corpus of single-typing errors.

Unigram Prior probability

Counts from 404,253,213 words in Corpus of Contemporary English (COCA)

word	Frequency of word	P(word)
actress	9,321	.0000230573
cress	220	.0000005442
caress	686	.0000016969
access	37,038	.0000916207
across	120,844	.0002989314
acres	12,874	.0000318463

Channel model probability

- **Error model probability, Edit probability**
- *Kernighan, Church, Gale 1990*

- *Misspelled word* $x = x_1, x_2, x_3 \ldots x_m$
- *Correct word* $w = w_1, w_2, w_3, \ldots, w_n$

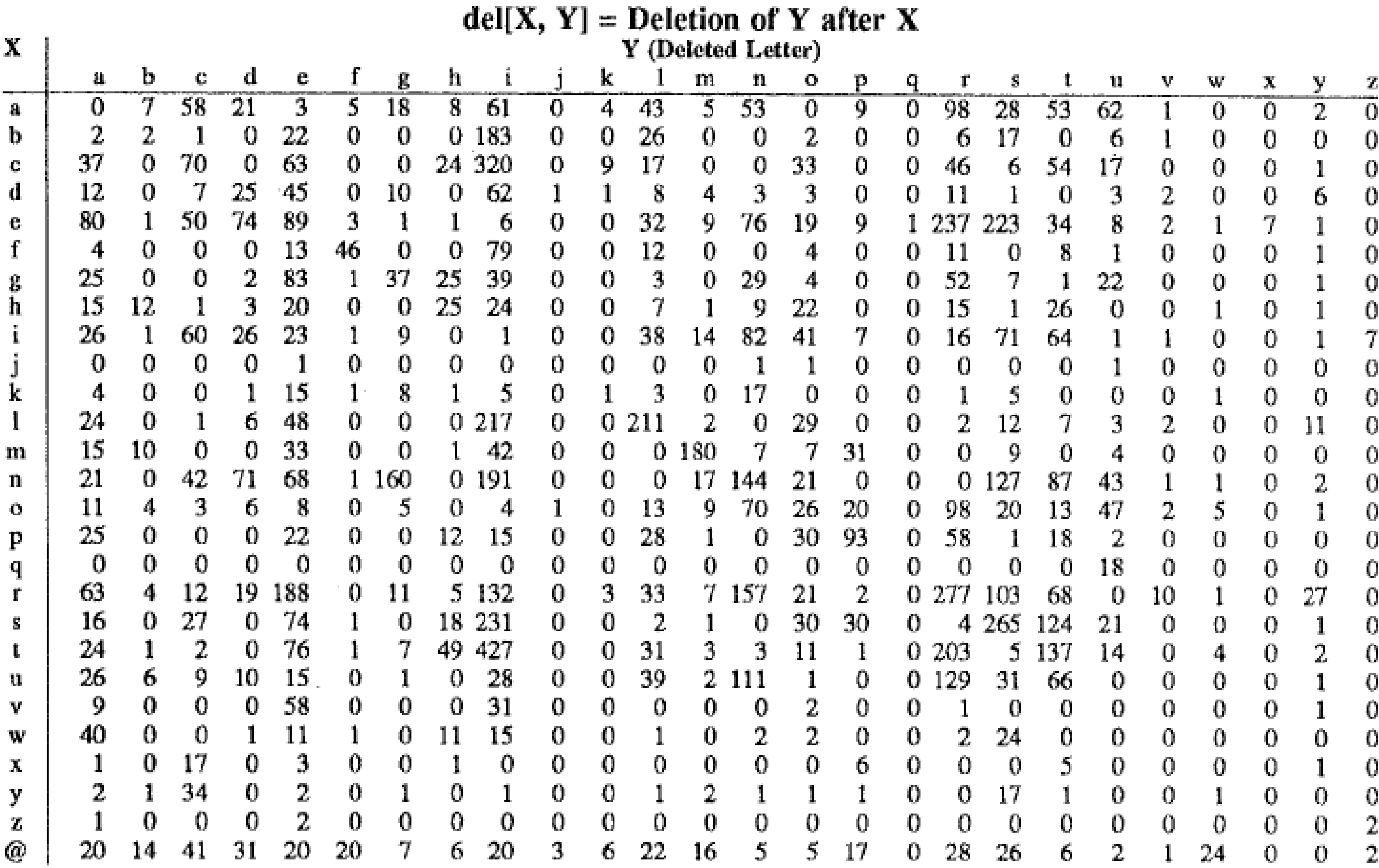
- $P(x|w)$ = probability of the edit
  - (deletion/insertion/substitution/transposition)

Computing error probability: confusion matrix

```
del[x,y]:    count(xy typed as x)
ins[x,y]:    count(x typed as xy)
sub[x,y]:    count(x typed as y)
trans[x,y]:  count(xy typed as yx)
```

Insertion and deletion conditioned on previous character

Confusion matrix – Deletion [Kerni90]

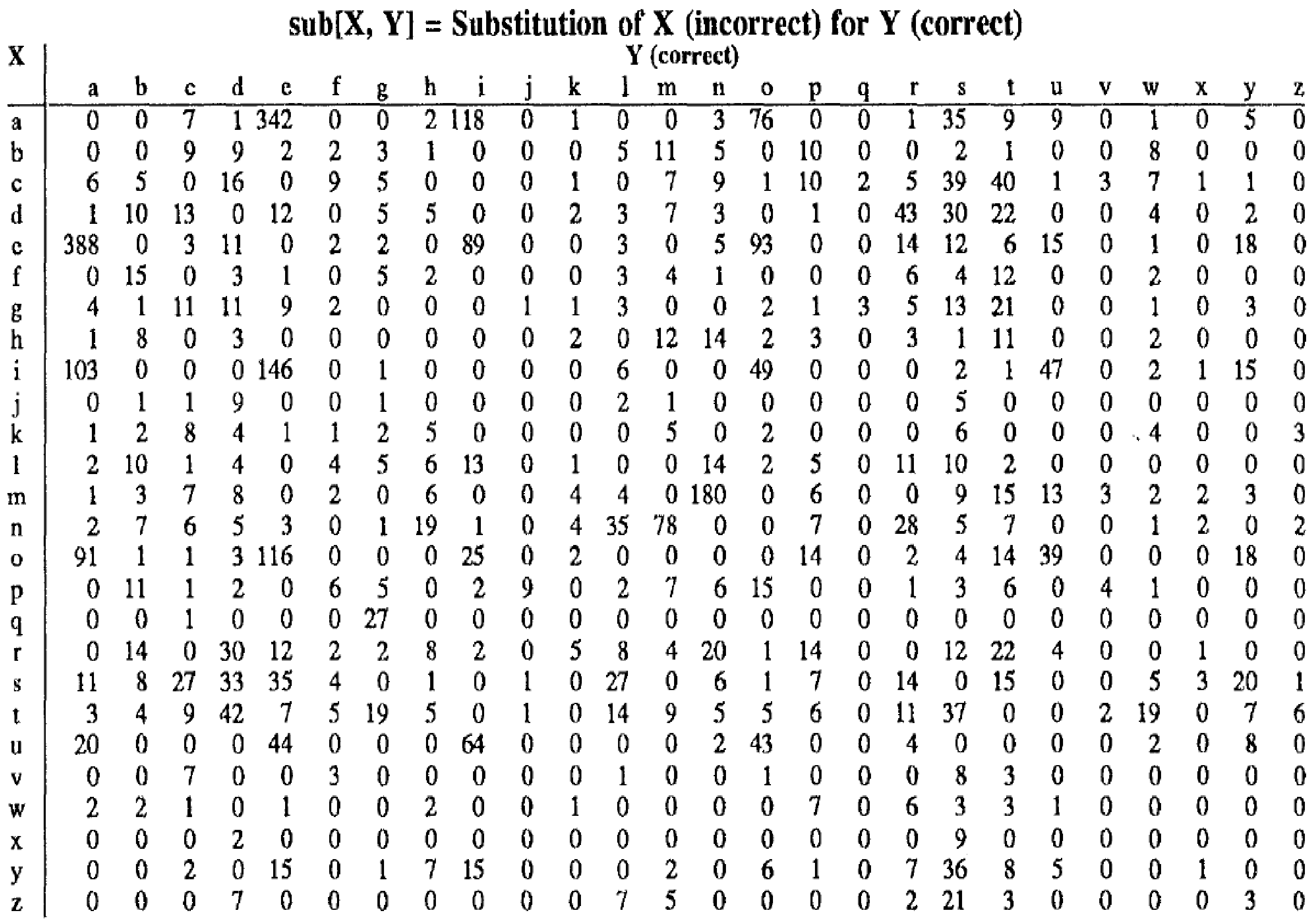


del[X, Y] = Deletion of Y after X

Y (Deleted Letter)

X	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	7	58	21	3	5	18	8	61	0	4	43	5	53	0	9	0	98	28	53	62	1	0	0	2	0
b	2	2	1	0	22	0	0	0	183	0	0	26	0	0	2	0	0	6	17	0	6	1	0	0	0	0
c	37	0	70	0	63	0	0	24	320	0	9	17	0	0	33	0	0	46	6	54	17	0	0	0	1	0
d	12	0	7	25	45	0	10	0	62	1	1	8	4	3	3	0	0	11	1	0	3	2	0	0	6	0
e	80	1	50	74	89	3	1	1	6	0	0	32	9	76	19	9	1	237	223	34	8	2	1	7	1	0
f	4	0	0	0	13	46	0	0	79	0	0	12	0	0	4	0	0	11	0	8	1	0	0	0	1	0
g	25	0	0	2	83	1	37	25	39	0	0	3	0	29	4	0	0	52	7	1	22	0	0	0	1	0
h	15	12	1	3	20	0	0	25	24	0	0	7	1	9	22	0	0	15	1	26	0	0	1	0	1	0
i	26	1	60	26	23	1	9	0	1	0	0	38	14	82	41	7	0	16	71	64	1	1	0	0	1	7
j	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0
k	4	0	0	1	15	1	8	1	5	0	1	3	0	17	0	0	0	1	5	0	0	0	1	0	0	0
l	24	0	1	6	48	0	0	0	217	0	0	211	2	0	29	0	0	2	12	7	3	2	0	0	11	0
m	15	10	0	0	33	0	0	1	42	0	0	0	180	7	7	31	0	0	9	0	4	0	0	0	0	0
n	21	0	42	71	68	1	160	0	191	0	0	0	17	144	21	0	0	0	127	87	43	1	1	0	2	0
o	11	4	3	6	8	0	5	0	4	1	0	13	9	70	26	20	0	98	20	13	47	2	5	0	1	0
p	25	0	0	0	22	0	0	12	15	0	0	28	1	0	30	93	0	58	1	18	2	0	0	0	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	0
r	63	4	12	19	188	0	11	5	132	0	3	33	7	157	21	2	0	277	103	68	0	10	1	0	27	0
s	16	0	27	0	74	1	0	18	231	0	0	2	1	0	30	30	0	4	265	124	21	0	0	0	1	0
t	24	1	2	0	76	1	7	49	427	0	0	31	3	3	11	1	0	203	5	137	14	0	4	0	2	0
u	26	6	9	10	15	0	1	0	28	0	0	39	2	111	1	0	0	129	31	66	0	0	0	0	1	0
v	9	0	0	0	58	0	0	0	31	0	0	0	0	0	2	0	0	1	0	0	0	0	0	0	1	0
w	40	0	0	1	11	1	0	11	15	0	0	1	0	2	2	0	0	2	24	0	0	0	0	0	0	0
x	1	0	17	0	3	0	0	1	0	0	0	0	0	0	0	6	0	0	0	5	0	0	0	0	1	0
y	2	1	34	0	2	0	1	0	1	0	0	1	2	1	1	1	0	0	17	1	0	0	1	0	0	0
z	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
@	20	14	41	31	20	20	7	6	20	3	6	22	16	5	5	17	0	28	26	6	2	1	24	0	0	2

Confusion matrix : substitution



sub[X, Y] = Substitution of X (incorrect) for Y (correct)

X	Y (correct)																									
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	7	1	342	0	0	2	118	0	1	0	0	3	76	0	0	1	35	9	9	0	1	0	5	0
b	0	0	9	9	2	2	3	1	0	0	0	5	11	5	0	10	0	0	2	1	0	0	8	0	0	0
c	6	5	0	16	0	9	5	0	0	0	1	0	7	9	1	10	2	5	39	40	1	3	7	1	1	0
d	1	10	13	0	12	0	5	5	0	0	2	3	7	3	0	1	0	43	30	22	0	0	4	0	2	0
e	388	0	3	11	0	2	2	0	89	0	0	3	0	5	93	0	0	14	12	6	15	0	1	0	18	0
f	0	15	0	3	1	0	5	2	0	0	0	3	4	1	0	0	0	6	4	12	0	0	2	0	0	0
g	4	1	11	11	9	2	0	0	0	1	1	3	0	0	2	1	3	5	13	21	0	0	1	0	3	0
h	1	8	0	3	0	0	0	0	0	0	2	0	12	14	2	3	0	3	1	11	0	0	2	0	0	0
i	103	0	0	0	146	0	1	0	0	0	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0
j	0	1	1	9	0	0	1	0	0	0	0	2	1	0	0	0	0	0	5	0	0	0	0	0	0	0
k	1	2	8	4	1	1	2	5	0	0	0	0	5	0	2	0	0	0	6	0	0	0	4	0	0	3
l	2	10	1	4	0	4	5	6	13	0	1	0	0	14	2	5	0	11	10	2	0	0	0	0	0	0
m	1	3	7	8	0	2	0	6	0	0	4	4	0	180	0	6	0	0	9	15	13	3	2	2	3	0
n	2	7	6	5	3	0	1	19	1	0	4	35	78	0	0	7	0	28	5	7	0	0	1	2	0	2
o	91	1	1	3	116	0	0	0	25	0	2	0	0	0	0	14	0	2	4	14	39	0	0	0	18	0
p	0	11	1	2	0	6	5	0	2	9	0	2	7	6	15	0	0	1	3	6	0	4	1	0	0	0
q	0	0	1	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	14	0	30	12	2	2	8	2	0	5	8	4	20	1	14	0	0	12	22	4	0	0	1	0	0
s	11	8	27	33	35	4	0	1	0	1	0	27	0	6	1	7	0	14	0	15	0	0	5	3	20	1
t	3	4	9	42	7	5	19	5	0	1	0	14	9	5	5	6	0	11	37	0	0	2	19	0	7	6
u	20	0	0	0	44	0	0	0	64	0	0	0	0	2	43	0	0	4	0	0	0	0	2	0	8	0
v	0	0	7	0	0	3	0	0	0	0	0	1	0	0	1	0	0	0	8	3	0	0	0	0	0	0
w	2	2	1	0	1	0	0	2	0	0	1	0	0	0	0	7	0	6	3	3	1	0	0	0	0	0
x	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0
y	0	0	2	0	15	0	1	7	15	0	0	0	2	0	6	1	0	7	36	8	5	0	0	1	0	0
z	0	0	0	7	0	0	0	0	0	0	0	7	5	0	0	0	0	2	21	3	0	0	0	0	3	0

Channel model

Kernighan, Church, Gale 1990

$$P(x|w) = \begin{cases} \dfrac{\mathrm{del}[w_{i-1}, w_i]}{\mathrm{count}[w_{i-1} w_i]}, & \text{if deletion} \\ \dfrac{\mathrm{ins}[w_{i-1}, x_i]}{\mathrm{count}[w_{i-1}]}, & \text{if insertion} \\ \dfrac{\mathrm{sub}[x_i, w_i]}{\mathrm{count}[w_i]}, & \text{if substitution} \\ \dfrac{\mathrm{trans}[w_i, w_{i+1}]}{\mathrm{count}[w_i w_{i+1}]}, & \text{if transposition} \end{cases}$$

Channel model for acress

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|word)
actress	t	–	c|ct	.000117
cress	–	a	a|#	.00000144
caress	ca	ac	ac|ca	.00000164
access	c	r	r|c	.000000209
across	o	e	e|o	.0000093
acres	–	s	es|e	.0000321
acres	–	s	ss|s	.0000342

Noisy channel probability for acress

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|word)	P(word)	10^9 *P(x|w)P(w)
actress	t	-	c|ct	.000117	.0000231	2.7
cress	-	a	a|#	.00000144	.000000544	.00078
caress	ca	ac	ac|ca	.00000164	.00000170	.0028
access	c	r	r|c	.000000209	.0000916	.019
across	o	e	e|o	.0000093	.000299	2.8
acres	-	s	es|e	.0000321	.0000318	1.0
acres	-	s	ss|s	.0000342	.0000318	1.0

Using a bigram language model

- □ "a stellar and versatile **acress** whose combination of sass and glamour…"
- □ Counts from the Corpus of Contemporary American English with add-1 smoothing
  - □ P(actress|versatile)=.000021
    P(whose|actress) = .0010
  - □ P(across|versatile) =.000021
    P(whose|across) = .000006

- □ **P("versatile actress whose") = .000021*.0010 = 210 x10^{-10}**
- □ P("versatile across whose") = .000021*.000006 = 1 x10^{-10}

Multiple Typing Errors

Multiple typing errors

- Measures of string similarity

  How similar is “intention” to “execution”?

- For strings of same length – Hamming distance

- Edit distance (A,B):

  minimum number of operations that transform string A into string B

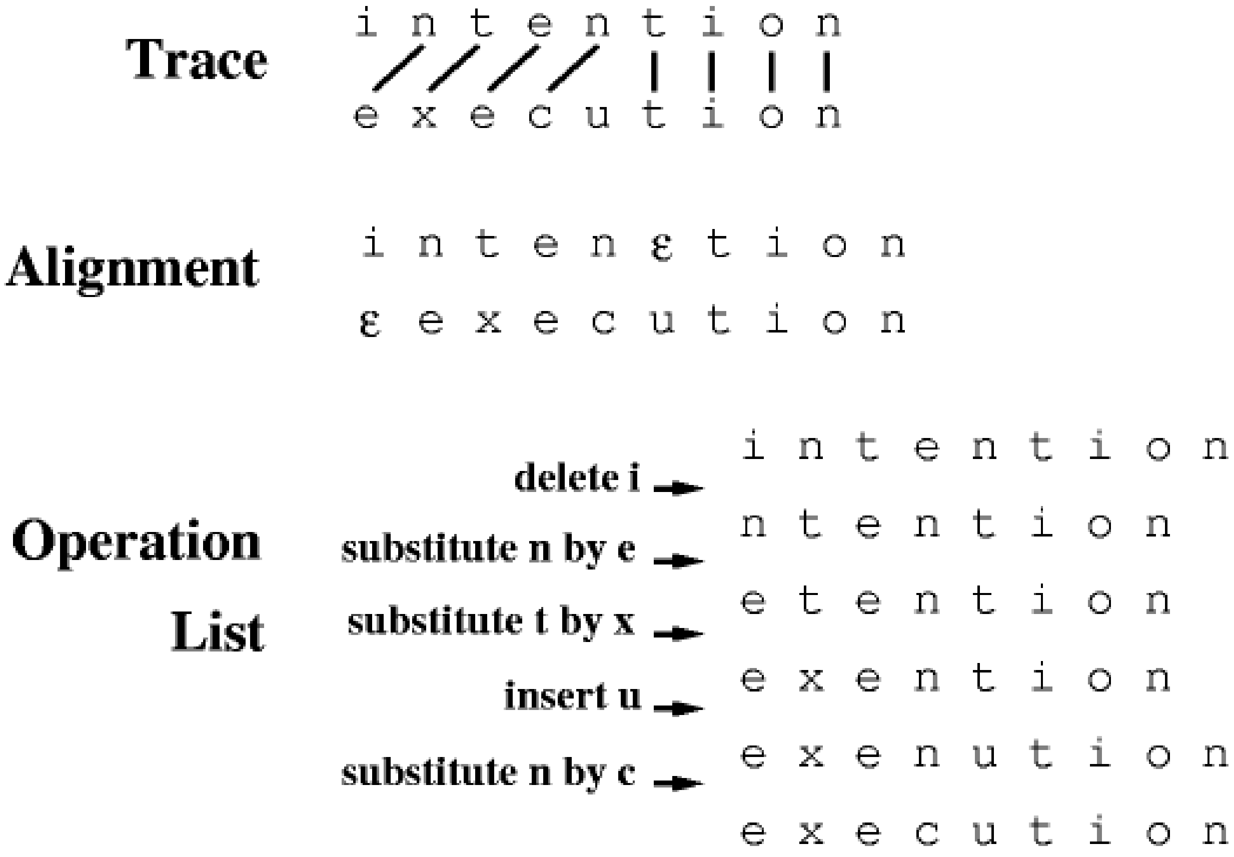
  - ins, del, sub, transp : Damerau –Levenshtein distance

Minimum Edit Distance

- Each edit operation has a cost
- Edit distance based measures
  - Levnishtein-Damreau distance

- How similar is “intension” to “execution”?

Three views of edit operations



Trace

intention

execution

Alignment

i	n	t	e	n	ε	t	i	o	n
ε	e	x	e	c	u	t	i	o	n

Operation List

intention

delete i → ntention

substitute n by e → etention

substitute t by x → exention

insert u → exenution

substitute n by c → execution

- □ All views →

  cost = 5 edits

- □ If subst / transp is not allowed [their cost = 2] →

  cost= 8 edits

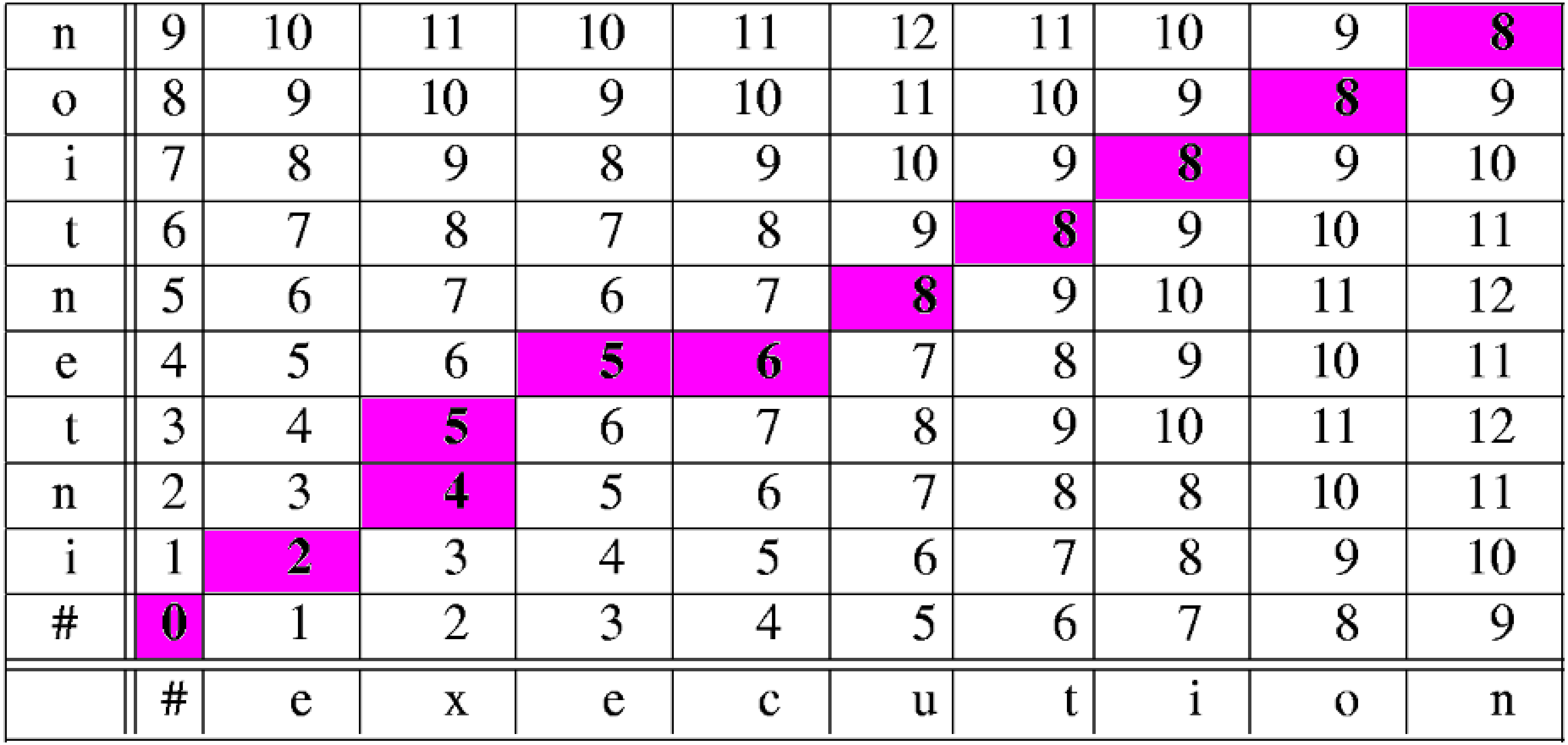
Levenshtein Distance

- len(A) = m; len (B) = n
- create n × m matrix : A along x-axis, B along y
- cost(i,j) = Levenshtein distance (A[0..i] , B[0..j])

  = cost of matching substrings

- Dynamic programming : solve by decomposition.
  - Dist-matrix(i,j) = min { costs of insert from (i-1,j) or (i,j-1); or cost of substitute from (i-1, j-1) }

Levenshtein Distance



n	9	10	11	10	11	12	11	10	9	**8**
o	8	9	10	9	10	11	10	9	**8**	9
i	7	8	9	8	9	10	9	**8**	9	10
t	6	7	8	7	8	9	**8**	9	10	11
n	5	6	7	6	7	**8**	9	10	11	12
e	4	5	6	**5**	**6**	7	8	9	10	11
t	3	4	**5**	6	7	8	9	10	11	12
n	2	3	**4**	5	6	7	8	8	10	11
i	1	**2**	3	4	5	6	7	8	9	10
#	**0**	1	2	3	4	5	6	7	8	9
	#	e	x	e	c	u	t	i	o	n

WORD-FROM-DICTIONARY SPELL CHECKER

Real-word spelling errors

- …leaving in about fifteen ***minuets*** to go to her house.
- The design ***an*** construction of the system…
- Can they ***lave*** him my messages?
- The study was conducted mainly ***be*** John Black.

- 25-40% of spelling errors are real words Kukich 1992

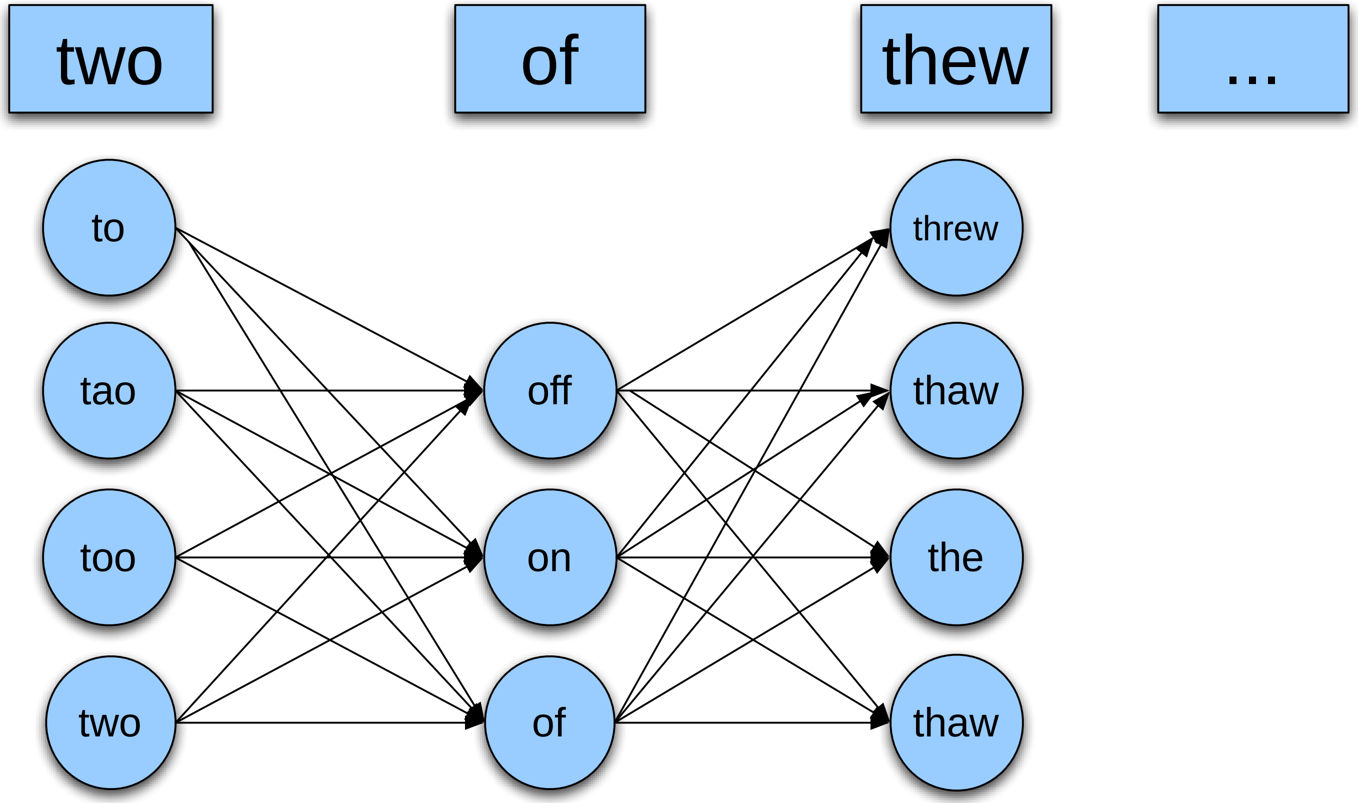
Solving real-world spelling errors

- For each word in sentence
  - Generate *candidate set*
    - the word itself
    - all single-letter edits that are English words
    - words that are homophones
- Choose best candidates
    - Noisy channel model
    - Task-specific classifier

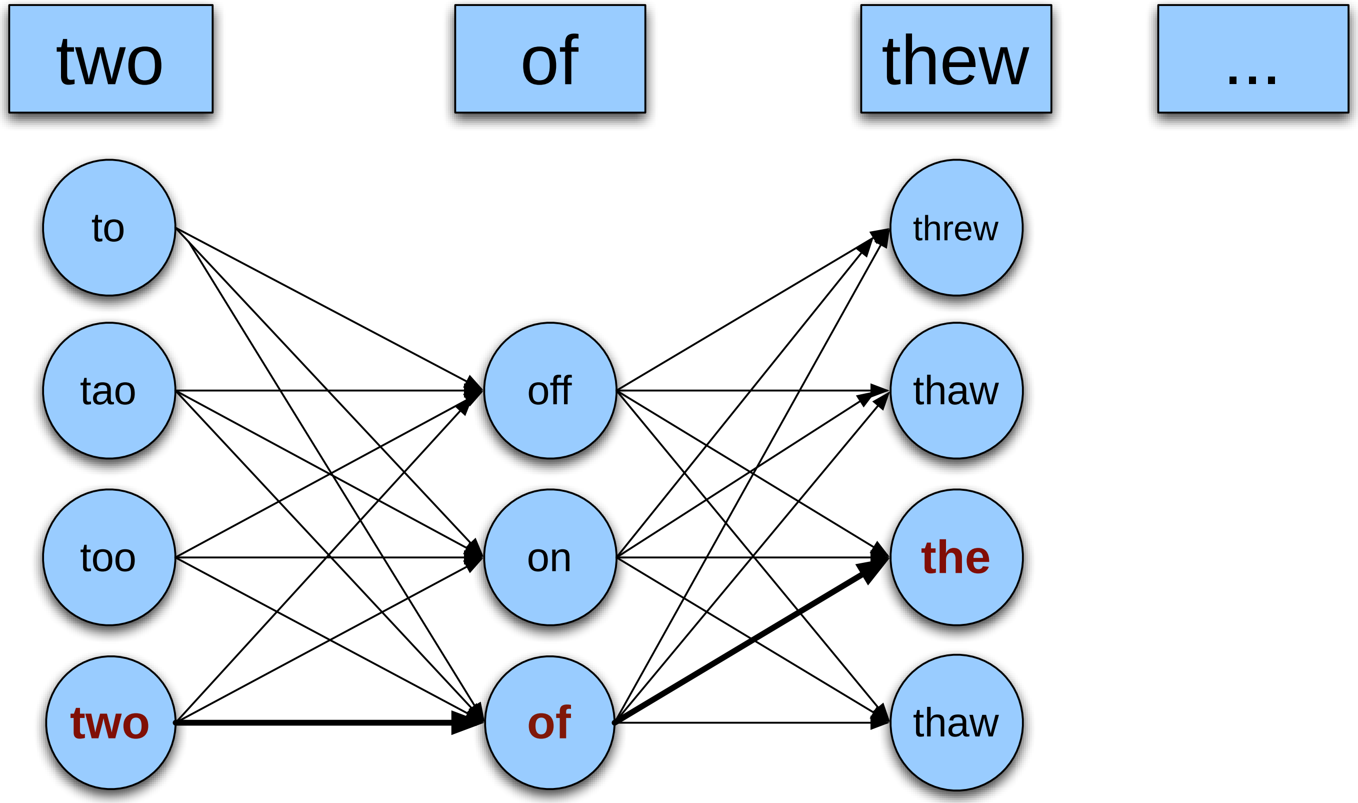
Noisy channel for real-word spell correction

- Given a sentence $w_1,w_2,w_3,\ldots,w_n$
- Generate a set of candidates for each word w_i
  - Candidate(w_1) = {w_1, w'_1 , w''_1 , w'''_1 ,…}
  - Candidate(w_2) = {w_2, w'_2 , w''_2 , w'''_2 ,…}
  - Candidate(w_n) = {w_n, w'_n , w''_n , w'''_n ,…}
- Choose the sequence W that maximizes P(W)

Noisy channel for real-word spell correction



Noisy channel for real-word spell correction



Norvig’s Python Spelling Corrector

How to Write a Spelling Corrector

http://norvig.com/spell-correct.html

Simplification: One error per sentence

- Out of all possible sentences with one word replaced
  - w_1, $\mathbf{w''_2}$,w_3,w_4 two **off** thew
  - w_1,w_2,$\mathbf{w'_3}$,w_4 two of **the**
  - $\mathbf{w'''_1}$,w_2,w_3,w_4 **too** of thew
  - …
- Choose the sequence W that maximizes P(W)

Where to get the probabilities

- Language model
  - Unigram
  - Bigram
  - Etc
- Channel model
  - Same as for non-word spelling correction
  - Plus need probability for no error, P(w|w)

Probability of no error

- What is the channel probability for a correctly typed word?
- P("the" | "the") = 1 – probability of mistyping

- Depends on typist, task, etc.
  - .90 (1 error in 10 words)
  - .95 (1 error in 20 words) ← value used, say
  - .99 (1 error in 100 words)
  - .995 (1 error in 200 words)

from http://norvig.com/ngrams/ch14.pdf p.235

Peter Norvig’s “thew” example

x	w	x|w	P(x|w)	P(w)	10^9 P(x|w)P(w)
thew	the	ew|e	0.000007	0.02	144
thew	thew		0.95	0.00000009	90
thew	thaw	e|a	0.001	0.0000007	0.7
thew	threw	h|hr	0.000008	0.000004	0.03
thew	thwe	ew|we	0.000003	0.00000004	0.0001

Choosing 0.99 instead of 0.95 (1 mistyping in 100 words) → ”thew” becomes more likely

State of the art noisy channel

- □ We never just multiply the prior and the error model
- □ Independence assumptions→probabilities not commensurate
- □ Instead: weight them

$$\hat{w} = \operatorname*{argmax}_{w \in V} P(x \mid w) P(w)^{\lambda}$$

- □ Learn λ from a validation test set
  (divide training set into training + validation)

Phonetic error model

- Metaphone, used in GNU aspell
  - Convert misspelling to metaphone pronunciation
    - “Drop duplicate adjacent letters, except for C.”
    - “If the word begins with 'KN', 'GN', 'PN', 'AE', 'WR', drop the first letter.”
    - “Drop 'B' if after 'M' and if it is at the end of the word”
    - …
  - Find words whose pronunciation is 1-2 edit distance from misspelling’s
  - Score result list
    - Weighted edit distance of candidate to misspelling
    - Edit distance of candidate pronunciation to misspelling pronunciation

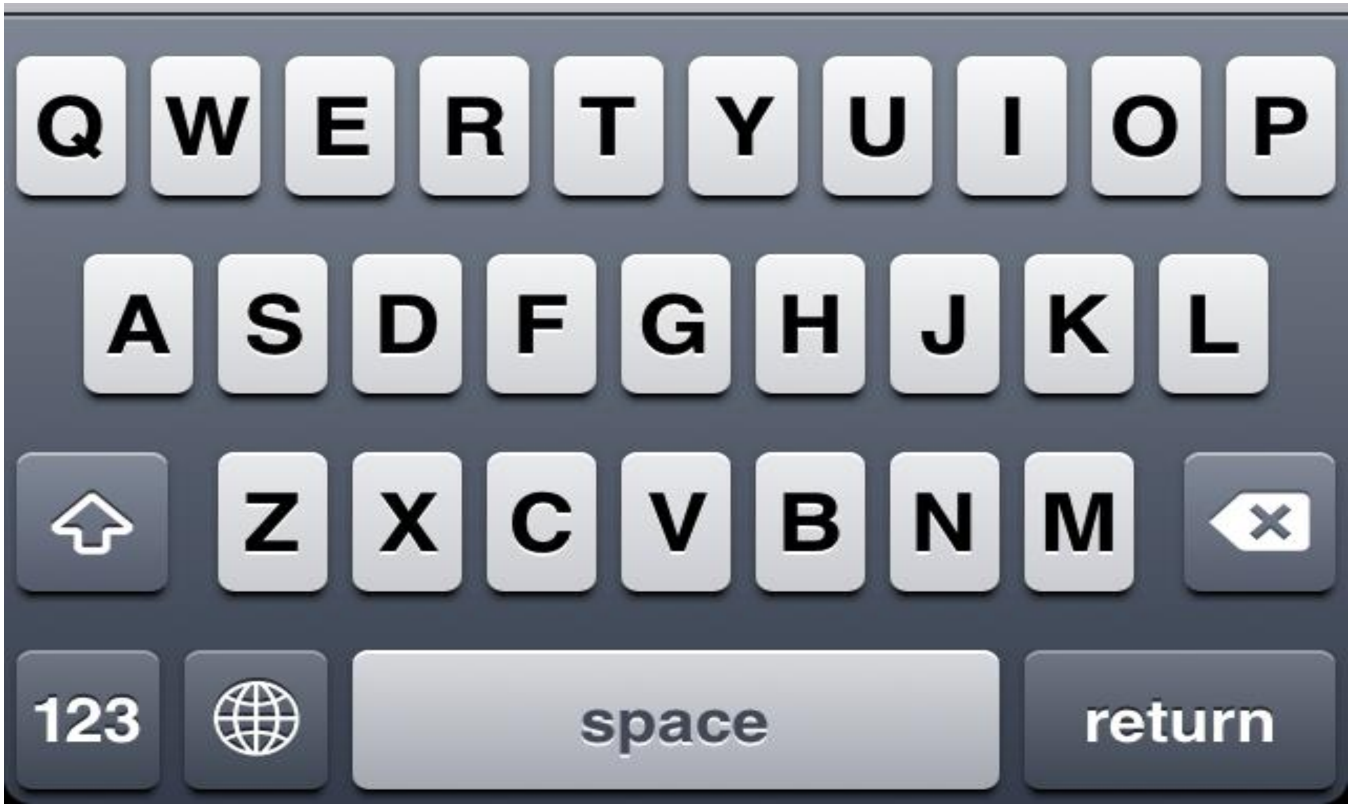
Improvements to channel model

- Allow richer edits (Brill and Moore 2000)
  - ent → ant
  - ph → f
  - le → al
- Incorporate pronunciation into channel (Toutanova and Moore 2002)

Channel model

- Factors that could influence p(misspelling|word)
  - The source letter
  - The target letter
  - Surrounding letters
  - The position in the word
  - Nearby keys on the keyboard
  - Homology on the keyboard
  - Pronunciations
  - Likely morpheme transformations

Nearby keys



Classifier-based methods

- Instead of just channel model and language model
- Use many more features – wider context
  build a classifier (machine learning).
- Example:

  whether/weather

  - "cloudy" within +- 10 words
  - ___ to VERB
  - ___ or not
- Q. How can we discover such features?

Candidate generation

- Words with similar spelling
  - Small edit distance to error
- Words with similar pronunciation
  - Small edit distance of pronunciation to error

Damerau-Levenshtein edit distance

- Minimal edit distance between two strings, where edits are:
  - Insertion
  - Deletion
  - Substitution
  - Transposition of two adjacent letters

Candidate generation

- 80% of errors are within edit distance 1
- Almost all errors within edit distance 2

- Also allow insertion of **space** or **hyphen**
  - `thisidea` → `this idea`
  - `inlaw` → `in-law`

Language Model

- Language modeling algorithms :
    - Unigram, bigram, trigram
    - Formal grammars
    - Probabilistic grammars

"You spelled garbage wrong."

CS 671 NLP NAÏVE BAYES AND SPELLING

amitabha mukerjee
iit kanpur

HCI issues in spelling

- If very confident in correction
  - Autocorrect
- Less confident
  - Give the best correction
- Less confident
  - Give a correction list
- Unconfident
  - Just flag as an error

Noisy channel based methods

- **IBM**
  - Mays, Eric, Fred J. Damerau and Robert L. Mercer. 1991. Context based spelling correction. *Information Processing and Management*, 23(5), 517–522
- **AT&T Bell Labs**
  - Kernighan, Mark D., Kenneth W. Church, and William A. Gale. 1990. A spelling correction program based on a noisy channel model. Proceedings of COLING 1990, 205-210