

NAIVE BAYES AND SENTIMENT CLASSIFICATION

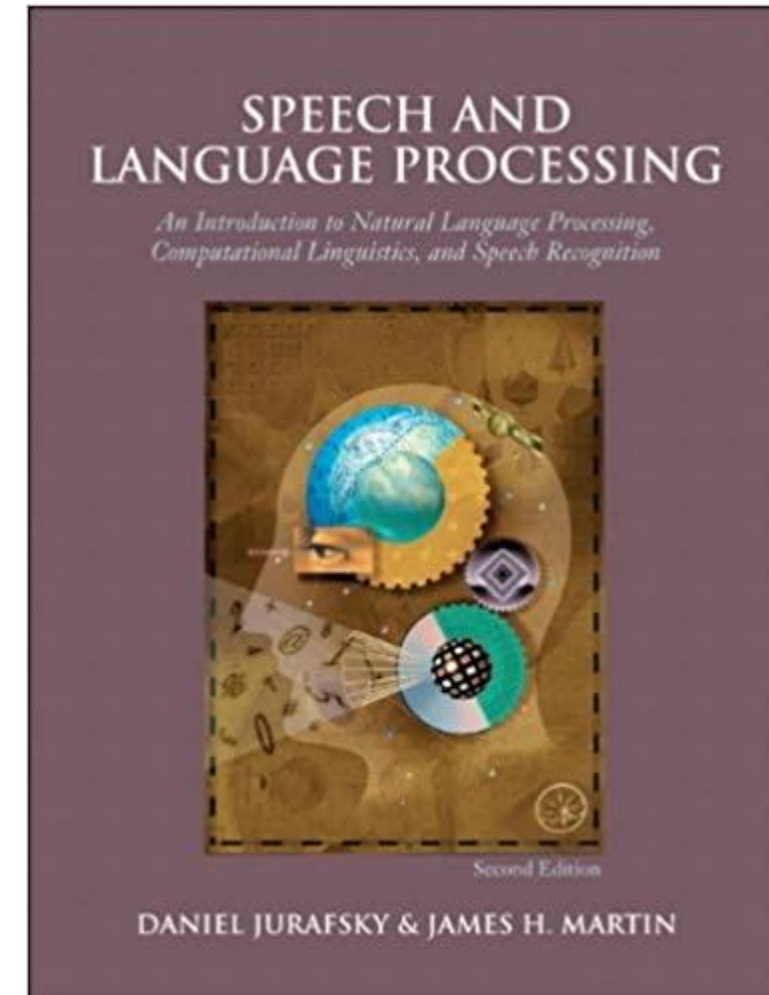
Spring 2023

CS6431 Natural Language Processing

B1:

*Speech and Language Processing (Third Edition draft
– Jan2022)*

Daniel Jurafsky, James H. Martin



Credits

1. B1

Assignment

Read:

B1: Chapter 4

Problems: Exercise problems of Chapter 4

Text Categorization

- Assigning a label/category to an entire sentence/document
- Sentiment analysis
 - ▣ Assigning a positive or negative orientation that a writer expresses toward some object.
 - ▣ Book reviews, movie reviews, product reviews etc.
- Spam detection
- Authorship attribution
- Subject category assignment

Examples of
text classification

Supervised Learning Approach

- Input:

- $(d_1, c_1), (d_2, c_2), \dots, (d_N, c_N)$
- And an unknown document d

- Output

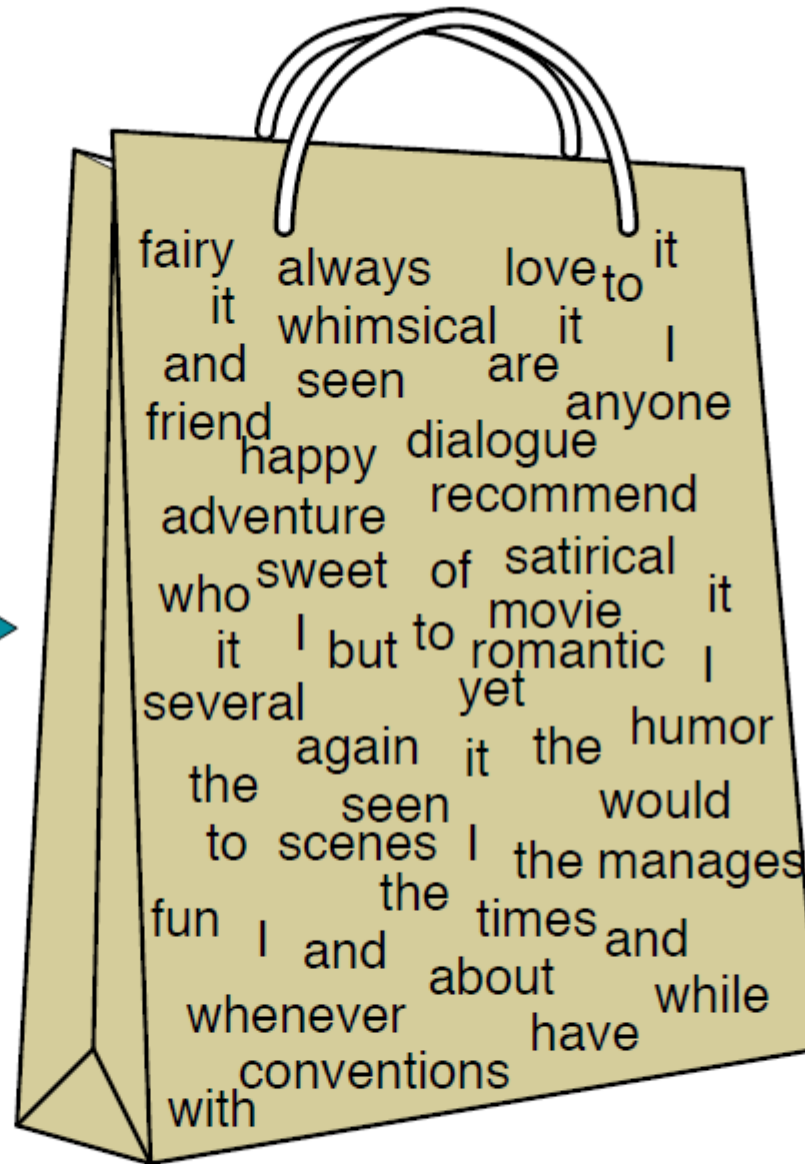
- The class label for d



Naive Bayes Classifiers

Bag-of-words

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!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

Position is ignored, only frequencies are used

Naïve Bayes

- Returns the class \hat{c} which has the maximum posterior probability given the document.

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

- Plugging the Bayes rule in the above $P(x|y) = \frac{P(y|x)P(x)}{P(y)}$

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

- Dropping denominator

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

$$\hat{c} = \operatorname{argmax}_{c \in C} \overbrace{P(d|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}$$

- Document d be represented as a set of features $\{f_1, f_2, \dots, f_n\}$

$$\hat{c} = \operatorname{argmax}_{c \in C} \overbrace{P(f_1, f_2, \dots, f_n|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}$$

□ Two simplifying assumptions

- ▣ Position of the word is not considered (does not matter)
- ▣ Naïve Bayes assumption All features are independent of each other

$$P(f_1, f_2, \dots, f_n | c) = P(f_1 | c) \cdot P(f_2 | c) \cdot \dots \cdot P(f_n | c)$$

□ The final equation

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

$$c_{NB} = \operatorname{argmax}_{c \in C} P(c) \prod_{i \in \text{positions}} P(w_i | c)$$

- The calculations are done in log space, to avoid underflow and increase speed

$$c_{NB} = \operatorname{argmax}_{c \in C} P(c) \prod_{i \in \text{positions}} P(w_i | c)$$

Becomes

$$c_{NB} = \operatorname{argmax}_{c \in C} \log P(c) + \sum_{i \in \text{positions}} \log P(w_i | c)$$

Training the Naive Bayes Classifier

- How to compute $P(c)$ and $P(w_i|c)$?

- $P(c) = \frac{N_c}{N_{doc}}$

 - ▣ N_c : number of documents labelled with c

 - ▣ N_{doc} : be the total number of documents

- We'll assume a feature is just the existence of a word in the document's bag of words

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

 - ▣ c : topic/class label

 - ▣ V : vocabulary of the dataset

A problem

- Consider the problem of movie reviews
- Imagine, no positive review in the training set contains “fantastic” but the test set does

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

- ▣ Probability for class “positive” will be zero
- Solution: Laplace (add-one) smoothing

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

- ▣ Note: vocabulary V consists of the union of all the word types in all classes, not just the words in one class c (why?)

More things to remove

- **Unknown Words:** words in test data but not in training data
 - ▣ **Ignore them** / remove them from test document/sentence
- **Stop words removal**
 - ▣ **Very frequent words** like 'the' and 'a'.
 - ▣ Sort by frequency and take top 10-100 entries as **stop words**
 - ▣ Or, use pre-defined list

function TRAIN NAIVE BAYES(D, C) **returns** $\log P(c)$ and $\log P(w|c)$

for each class $c \in C$ # Calculate $P(c)$ terms

N_{doc} = number of documents in D

N_c = number of documents from D in class c

$logprior[c] \leftarrow \log \frac{N_c}{N_{doc}}$

$V \leftarrow$ vocabulary of D

$bigdoc[c] \leftarrow$ **append**(d) **for** $d \in D$ **with** class c

for each word w in V # Calculate $P(w|c)$ terms

$count(w, c) \leftarrow$ # of occurrences of w in $bigdoc[c]$

$loglikelihood[w, c] \leftarrow \log \frac{count(w, c) + 1}{\sum_{w' \in V} (count(w', c) + 1)}$

return $logprior, loglikelihood, V$

function TEST NAIVE BAYES($testdoc, logprior, loglikelihood, C, V$) **returns** best c

for each class $c \in C$

$sum[c] \leftarrow logprior[c]$

for each position i in $testdoc$

$word \leftarrow testdoc[i]$

if $word \in V$

$sum[c] \leftarrow sum[c] + loglikelihood[word, c]$

return $\operatorname{argmax}_c sum[c]$



$$P(c) = \frac{N_c}{N_{doc}}$$

$$\hat{P}(w_i|c)$$

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

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

Improvements

- For a text classification task, whether a word occurs or not seems to matter more than its frequency
 - ▣ Clip the word counts in each document at 1
 - ▣ Binary Naïve Bayes

Four original documents:

- it was pathetic the worst part was the boxing scenes
- no plot twists or great scenes
- + and satire and great plot twists
- + great scenes great film

After per-document binarization:

- it was pathetic the worst part boxing scenes
- no plot twists or great scenes
- + and satire great plot twists
- + great scenes film

	NB Counts		Binary Counts	
	+	–	+	–
and	2	0	1	0
boxing	0	1	0	1
film	1	0	1	0
great	3	1	2	1
it	0	1	0	1
no	0	1	0	1
or	0	1	0	1
part	0	1	0	1
pathetic	0	1	0	1
plot	1	1	1	1
satire	1	0	1	0
scenes	1	2	1	2
the	0	2	0	1
twists	1	1	1	1
was	0	2	0	1
worst	0	1	0	1

Counts can be greater than 1 in Binary NB!

□ Dealing with negation

- ▣ I really like this movie (+ve)

- ▣ I didn't like this movie (-ve)

- Negation alters the meaning of every word

didn't like this movie , but I

becomes

didn't NOT_like NOT_this NOT_movie , but I

Naive Bayes

very fast

less storage

work well with small training data

robust to irrelevant features

very good in domains with many equally importance features

optimal if the independence assumptions hold

□ Insufficient training data

- ▣ Inaccurate training using Naïve Bayes

- ▣ Derive features using **sentiment lexicons**

 - Lists of words that are pre-annotated with positive or negative sentiment.

+ : *admirable, beautiful, confident, dazzling, ecstatic, favor, glee, great*

– : *awful, bad, bias, catastrophe, cheat, deny, envious, foul, harsh, hate*

 - Add a feature for +ve and –ve

 - Count of +ve/-ve feature = count of words from the corresponding lexicon

Naive Bayes as a Language Model

- Assigns probability to N-grams and sentences, hence it can also be seen as a Language Model

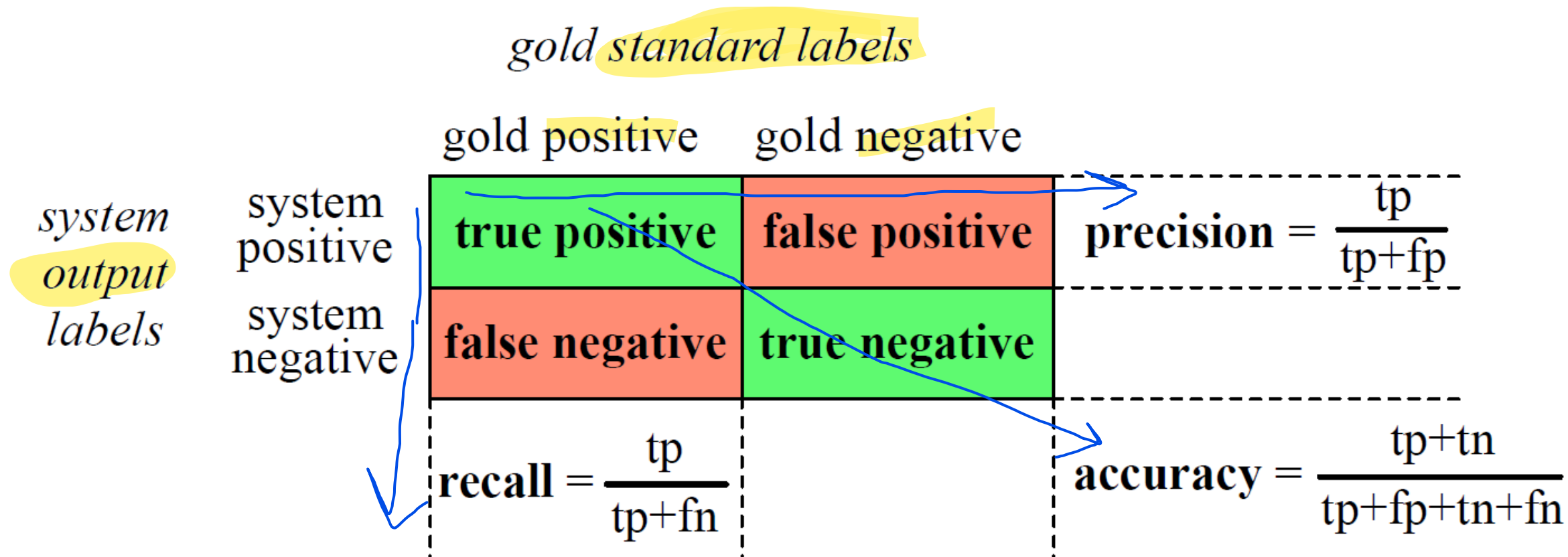
EACH CLASS = CLASS CONDITIONED UNIGRAM MODEL




Evaluation Metrics



Confusion Matrix




$$\mathbf{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

$$\mathbf{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

- Precision and Recall alone are not sufficient (why?)

F-measure

$$F_{\beta} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

$\beta > 1$ favors recall

$\beta < 1$ favors precision

$\beta = 1$ equal importance to precision and recall

□ $F_{\beta=1}$ or $F_1 = \frac{2PR}{P+R}$

▣ Harmonic mean is more conservative than arithmetic mean

■ Closer to the smaller of the two numbers

Evaluating more than two classes

		<i>gold labels</i>			
		urgent	normal	spam	
<i>system output</i>	urgent	8	10	1	$\text{precision}_u = \frac{8}{8+10+1}$
	normal	5	60	50	$\text{precision}_n = \frac{60}{5+60+50}$
	spam	3	30	200	$\text{precision}_s = \frac{200}{3+30+200}$
		$\text{recall}_u = \frac{8}{8+5+3}$	$\text{recall}_n = \frac{60}{10+60+30}$	$\text{recall}_s = \frac{200}{1+50+200}$	

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

Pooled

	true yes	true no
system yes	268	99
system no	99	635

$$\text{microaverage
precision} = \frac{268}{268+99} = .73$$

$$\text{macroaverage
precision} = \frac{.42+.52+.86}{3} = .60$$

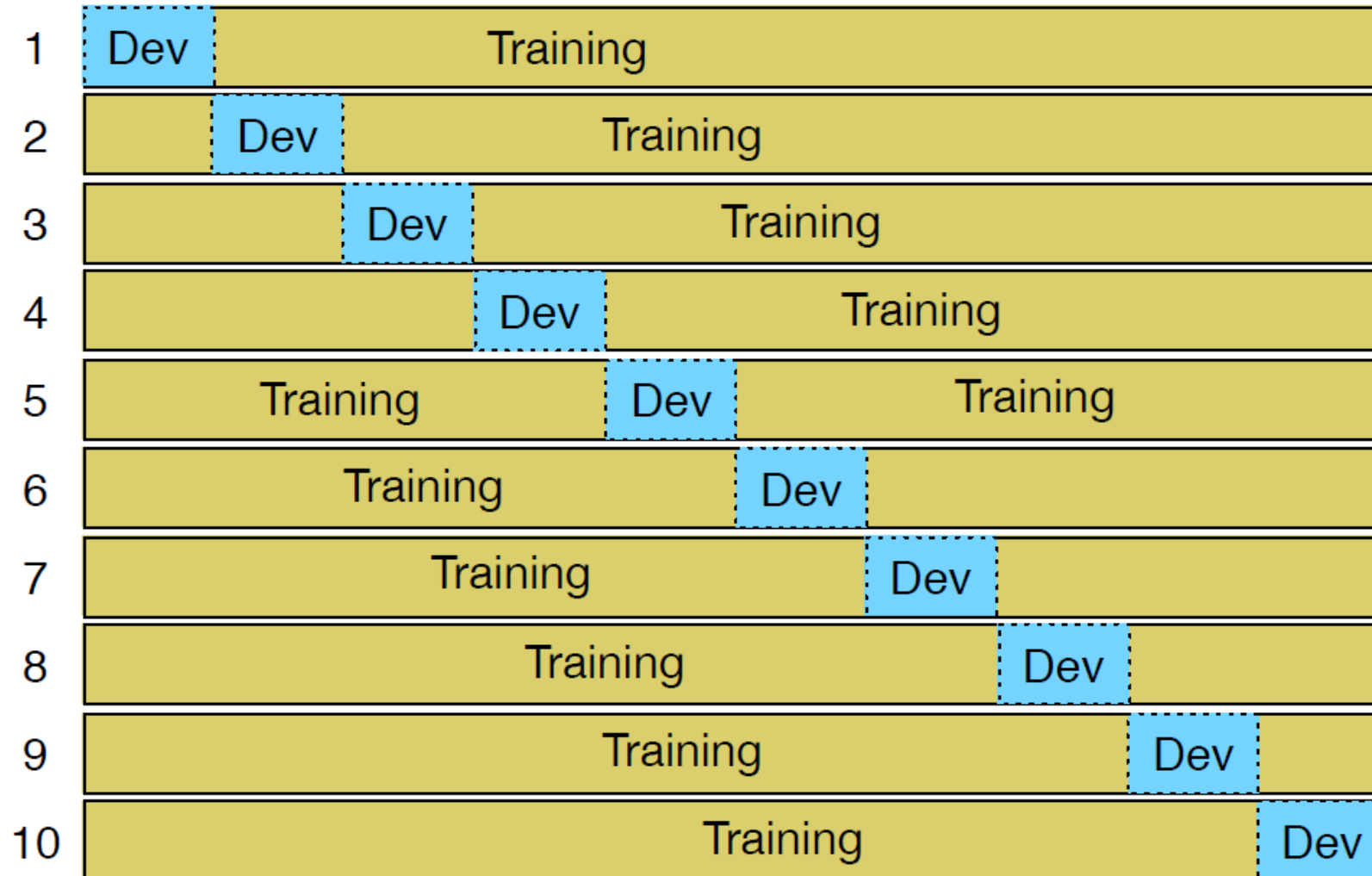
k -fold Cross Validation

unseen test set:
avoid overfitting
more conservative estimate of performance

cross validation handle sampling error

Training Iterations

Testing



Test Set

Statistical Significance Testing

- How to decide if model/classifier A is better than B ?
- $M(A, x)$: performance of model/classifier A on test set x
- $M(B, x)$: performance of model/classifier B on test set x
 $\delta(x)$ (effect size) = $M(A, x) - M(B, x)$
- Consider $\delta(x) = .04$
- We want to check if A 's superiority over B is likely to hold again if we checked another test set x'
- We define two hypothesis
 - ▣ H_0 : $\delta(x) \leq 0$ (Null Hypothesis, A is not better than B)
 - ▣ H_1 : $\delta(x) > 0$ (A is better than B)

- $H_0: \delta(x) \leq 0$ (Null Hypothesis, A is not better than B)
 - ▣ We want to test if we can confidently rule out the null hypothesis and instead support H_1 , i.e., A is better
- Let X : R.V. over all test sets

$$P(\delta(X) \geq \delta(x) | H_0 \text{ is true})$$

- ▣ **p-value**: the probability, assuming the null hypothesis H_0 is true, of seeing the $\delta(x)$ that we saw or one even greater
- ▣ If H_0 is indeed true
 - Large $\delta(x)$: highly surprising, p-value should be low, reject null hypothesis
 - Small (+ve) $\delta(x)$: less surprising even if H_0 is true, p-value should be high
- ▣ Threshold (like .01)
 - p-value < .01, reject null hypothesis
- ▣ We say that a result (e.g., “ A is better than B ”) is statistically significant if the δ we saw has a probability that is below the threshold and we therefore reject this null hypothesis.

- How to estimate P -values?
 - ▣ Create multiple test sets and measure
 - ▣ Use a threshold to accept/reject a hypothesis

The Paired Bootstrap Test

- Bootstrapping: repeatedly drawing large numbers of smaller samples with replacement

	1	2	3	4	5	6	7	8	9	10	A%	B%	$\delta()$
x	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	.70	.50	.20
$x^{(1)}$	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	.60	.60	.00
$x^{(2)}$	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	.60	.70	-.10
...	$\text{p-value}(x) = \frac{1}{b} \sum_{i=1}^b \mathbb{1} \left(\delta(x^{(i)}) - \delta(x) \geq 0 \right)$												
$x^{(b)}$													

$\mathbb{1}(x)$: if x is true, and 0 otherwise

Distribution of
 δ values

	1	2	3	4	5	6	7	8	9	10	A%	B%	$\delta()$
x	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	.70	.50	.20
$x^{(1)}$	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	.60	.60	.00
$x^{(2)}$	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	.60	.70	-.10
...													
$x^{(b)}$	$\text{p-value}(x) = \frac{1}{b} \sum_{i=1}^b \mathbb{1} \left(\delta(x^{(i)}) - \delta(x) \geq 0 \right)$												

- Goal: assume H_0 and estimate how accidental/surprising $\delta(x)$ is
- Since the above distribution is biased towards $\delta(x) = .2$, to capture how surprising $\delta(x)$ is, we compute:

$$\begin{aligned}
 \text{p-value}(x) &= \frac{1}{b} \sum_{i=1}^b \mathbb{1} \left(\delta(x^{(i)}) - \delta(x) \geq \delta(x) \right) \\
 &= \frac{1}{b} \sum_{i=1}^b \mathbb{1} \left(\delta(x^{(i)}) \geq 2\delta(x) \right)
 \end{aligned}$$

$$\begin{aligned}\text{p-value}(x) &= \frac{1}{b} \sum_{i=1}^b \mathbb{1} \left(\delta(x^{(i)}) - \delta(x) \geq \delta(x) \right) \\ &= \frac{1}{b} \sum_{i=1}^b \mathbb{1} \left(\delta(x^{(i)}) \geq 2\delta(x) \right)\end{aligned}$$

- Suppose,
 - ▣ 10,000 bootstrapped test sets ($x^{(i)}$ s) are created
 - ▣ Threshold is .01
 - ▣ Above gives p-value of .0047 (< threshold)
 - Thus, reject the null hypothesis

function BOOTSTRAP(test set x , num of samples b) **returns** $p\text{-value}(x)$

Calculate $\delta(x)$ # how much better does algorithm A do than B on x

$s = 0$

for $i = 1$ **to** b **do**

for $j = 1$ **to** n **do** # Draw a bootstrap sample $x^{(i)}$ of size n

 Select a member of x at random and add it to $x^{(i)}$

 Calculate $\delta(x^{(i)})$ # how much better does algorithm A do than B on $x^{(i)}$

$s \leftarrow s + 1$ **if** $\delta(x^{(i)}) \geq 2\delta(x)$

$p\text{-value}(x) \approx \frac{s}{b}$ # on what % of the b samples did algorithm A beat expectations?

return $p\text{-value}(x)$ # if very few did, our observed δ is probably not accidental

A version of the paired bootstrap algorithm after [Berg-Kirkpatrick et al. \(2012\)](#).