BUS 243

Lecture 3: Classification Primer

WHAT IS A SPAM?

- Consider how do you identify a spam email
  - What's your rule?
- …*we have a strong spam filtering system…*
  - What does it mean?

- Text classification (categorization)
  - Assign a label or category to corpus or document
    - Spam detection
    - Sentiment analysis
    - Language ID
    - Topic modeling

- Some words are indicative
  - Emotion, mood, attitudes: angry, nervous, great
  - Unique signal: Michael Jordan
  - Domain words: Monetary policy, Alzheimer's
- Any concerns?
- Build some rules
  - Can be fragile

- Focus on supervised machine learning
- Simple sentiment analysis task
  - is the attitude of this text positive or negative

DEFINITION

- Training set of N documents: $\{(d_1, c_1), \ldots, (d_N, c_N)\}$
  - document d (input) and a fixed set of output classes $C = \{c_1, \ldots, c_M\}$
- Goal is to learn a classifier capable of mapping from a new document d to its correct class $c \in C$
  - C could be a real number between 0 and 1, the probability of the observation being in the class

- Today, we study **generative classifiers** like naïve Bayes
- Next class, we turn to **discriminative classifiers**

NAÏVE BAYES

- Simple Naïve Bayes classifier
  - Rely on simplifying assumptions and very simple representation of document: **Bag of words**
- For a document d and class c
  - $\hat{c} = \text{argmax}_{c \in C} P(c|d)$

- $\hat{c} = \text{argmax}_{c \in C} P(c|d)$
  - Estimate the correct class having the maximum posterior probability given d (most likely class)
- Applying Bayes Rule
  - $\hat{c} = \text{argmax}_{c \in C} P(d|c)P(c)$
  - $\hat{c} = \text{argmax}_{c \in C} P(x_1, \ldots, x_n|c)P(c)$
    - Hard to compute: $O(|X|^n \cdot |C|)$ parameters

- Bag of Words assumptions: ignore the word positions

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
…	…

- Naïve Bayes assumption: conditional independence
  - $P(x_1, \ldots, x_n|c) = P(x_1|c) \cdot P(x_2|c) \cdots P(x_n|c)$
- Hence,
- $c_{MAP} = \text{argmax}_{c \in C} P(x_1, \ldots, x_n|c) P(c)$
- $c_{NB} = \text{argmax}_{c \in C} P(c_j) \Pi_{x \in X} P(x|c)$
- $c_{NB} = \text{argmax}_{c_j \in C} P(c_j) \Pi_{i \in \text{positions}} P(x_i|c_j)$
- Multiplying lots of probabilities can result in floating-point underflow

- $c_{NB} = \text{argmax}_{c_j \in C} [\log P(c_j) + \sum \log P(x_i | c_j)]$
- Taking log doesn't change the ranking of classes
- Linear classifiers
  - Use a linear combination of the inputs to make a classification decision

TRAINING THE CLASSIFIER

- Need to learn the probabilities $P(c)$ and $P(x_i|c)$
- Maximum likelihood estimate: use the frequencies in the training
  - $\hat{P}(c_j) = \frac{N_{c_j}}{N_{\text{total}}}$
  - $\hat{P}(w_i|c_j) = \frac{\text{count}(w_i,c_j)}{\sum_{w \in V} \text{count}(w,c_j)}$
    - Fraction of times word w_i appears among all words in documents of topic c_j
    - Vocabulary V consists of the union of all the word types in all classes, not just the words in one class c

- What if we have seen no training documents with the word *fantastic* and classified in the topic *positive*?
  - $\hat{P}(\text{“fantastic”}|\text{positive}) = 0$
  - Zero probabilities cannot be conditioned away , no matter the other evidence!
- Hence, we need to apply smoothing

- **Laplace smoothing**
  - $\hat{P}(w_i|c) = \frac{\text{count}(w_i,c_j)+a}{\sum_{w\in V}(\text{count}(w,c_j)+a)} = \frac{\text{count}(w_i,c_j)+a}{\sum_{w\in V}\text{count}(w,c_j)+a|V|}$
  - $a = 1$
- **Try to explain the following again in words**
  - $\hat{P}(c_j) = \frac{N_{c_j}}{N_{\text{total}}}$
  - $\hat{P}(w_i|c_j) = \frac{\text{count}(w_i,c_j)}{\sum_{w\in V}\text{count}(w,c_j)}$

UNKNOWN WORDS

- Some words appear in test data but not in training data (or vocabulary)
- Ignore them
  - Remove them from the test document
  - Don't include any probability for them at all
- Why don't we build an unknown word model?
  - It doesn't help
  - Knowing which class has more unknown words is not generally helpful

STOP WORDS

- Some systems ignore stop words
  - What's the stop words?
    - Very frequent words like *the* and *a*
  - Call the top 10 or 50 words the **stopword list**
  - Remove all stop words from both training and test sets
- However, removing stop words doesn't usually help
  - So in practice most NB algorithms use all words and don't use stopword lists
- When do stop words play a role?

EXAMPLE

	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

- What's the goal?

- Construct naïve Bayes with add-one smoothing
- The prior: $P(-) = \frac{3}{5}$ and $P(+) = \frac{2}{5}$
- Likelihoods from training: $\frac{\text{count}(w_i, c_j)+1}{\sum_{w \in V} \text{count}(w, c_j)+|V|}$
  - $P(\text{“predictable”}|-) = \frac{1+1}{14+20}, P(\text{“predictable”}|+) = \frac{0+1}{9+20}$
  - $P(\text{“no”}|-) = \frac{1+1}{14+20}, P(\text{“no”}|+) = \frac{0+1}{9+20}$
  - $P(\text{“fun”}|-) = \frac{0+1}{14+20}, P(\text{“fun”}|+) = \frac{1+1}{9+20}$

- For the test sentence $S =$ "predictable with no fun" after removing the word '*with*'
  - $P(-)P(S|-) = \frac{3}{5} \times \frac{2}{34} \times \frac{2}{34} \times \frac{1}{34} = 6.1 \times 10^{-5}$
  - $P(+)P(S|+) = \frac{2}{5} \times \frac{1}{29} \times \frac{1}{29} \times \frac{2}{29} = 3.2 \times 10^{-5}$
- The model thus predicts the class negative

SUMMARY: NB IS NOT SO NAÏVE

- Very fast, low storage requirements
- Work well with very small amounts of training data
- Robust to irrelevant features
  - Irrelevant features cancel each other without affecting results
- Very good in domains with many equally important features
- Optimal if the independence assumption hod
  - If so, then it is the bayes Optimal classifier for problem
- A good dependable baseline for text classification
  - But we will see other classifiers that give better accuracy

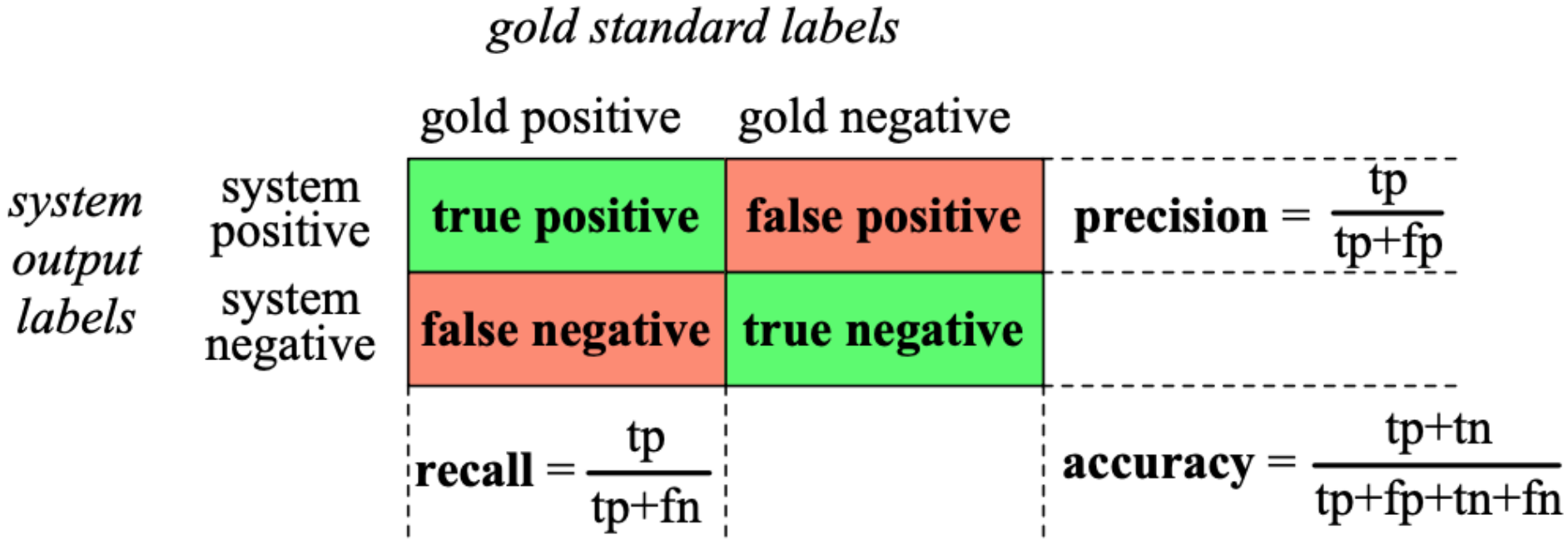
GENERATIVE MODEL

- $P(\text{“I love this fun film”}|+) > P(\text{“I love this fun film”}|-)$
  - What does it mean?
- Naïve Bayes has an important similarity to language modeling
- Consider a naïve Bayes model with the classes + and – and following params

w	P(w|+)	P(w|-)
I	0.1	0.2
love	0.1	0.001
this	0.01	0.01
fun	0.05	0.005
film	0.1	0.1
…	…	…

EVALUATION

- Need a metric for knowing how well our model is doing
- Revisit
  - Precision: What fraction of the returned results are relevant to the information need?
  - Recall: What fraction of the relevant documents in the collection were returned by the system?
- Consider building a confusion matrix



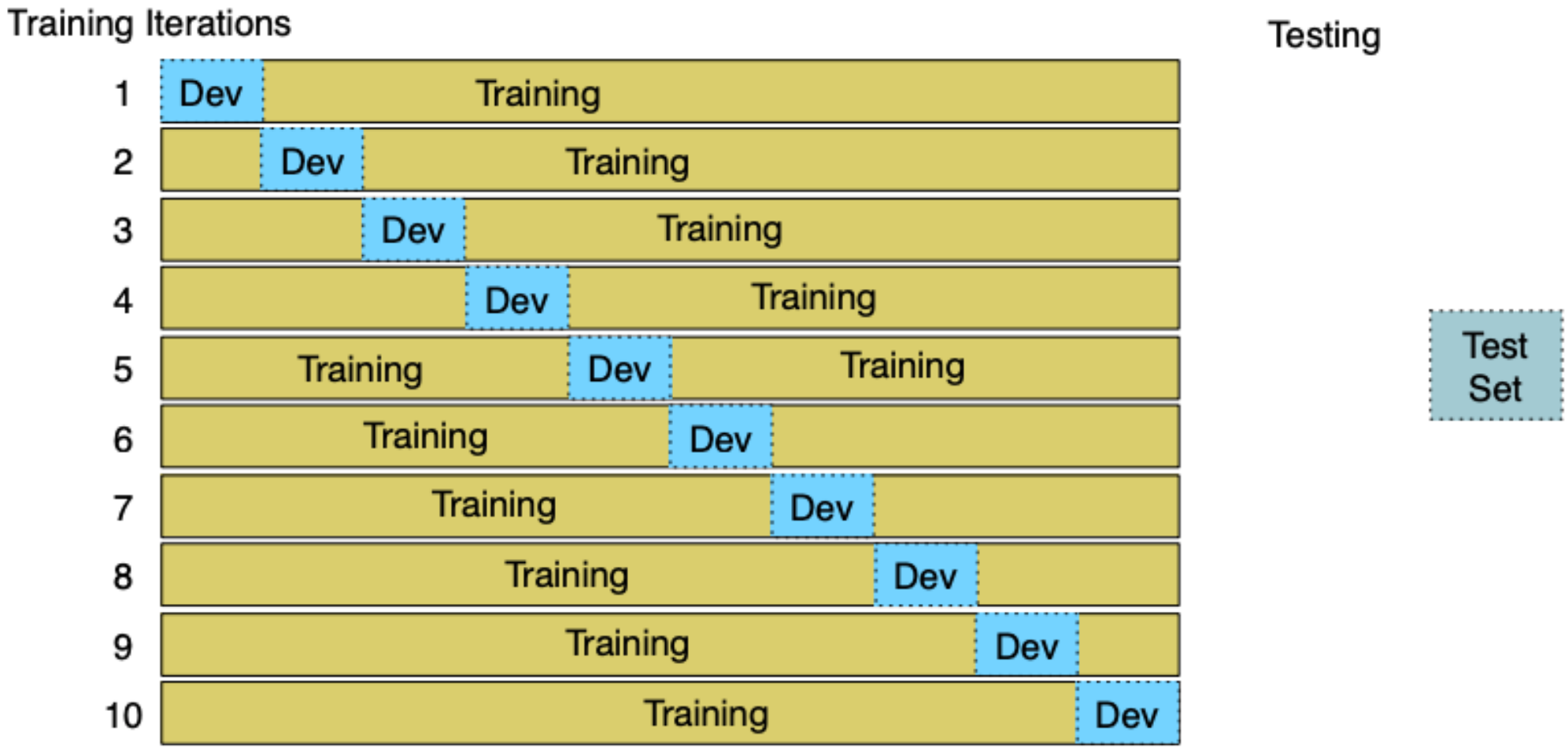
- Why don't we use accuracy as our metric?

- Precision: % of the items that the system labeled as positive that are in fact positive
  - $P = \frac{\text{true positives}}{\text{true positives+false positives}}$
- Recall: % of items actually present in the input that were correctly identified by the system
  - $R = \frac{\text{true positives}}{\text{true positives+false negatives}}$
- See the trade-off between them
- Precision and recall, unlike accuracy, emphasize true positives

- F-measure: combines P and R
  - $F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P+R}$
  - $\beta > 1$ favor recall, while $\beta < 1$ favor precision
  - $\beta = 1$ balances them and most popular
    - $F_1 = \frac{2PR}{P+R}$

VALIDATION

- Supply your intuition on
  - training set, development set, and test set
  - Cross-validation
    - K-fold cross validation


Training Iterations
Testing
1
Dev
Training
2
Dev
Training
3
Dev
Training
4
Dev
Training
5
Training
Dev
Training
6
Training
Dev
7
Training
Dev
8
Training
Dev
9
Training
Dev
10
Training
Dev
Test Set

- How do we know if one classifier is better than another?
  - Suppose Classifier A and B
  - $M(A,x)$ is the performance of A on test set x
  - $\delta(x) = M(A,x) - M(B,x)$
- Want to know if $\delta(x) > 0$ (A is better than B)
  - $\delta(x)$ is called the effect size
- Suppose we have $\delta(x) > 0$. Are we done?

STATISTICAL HYPOTHESIS TESTING

- $H_0: \delta(x) \leq 0$
- $H_1: \delta(x) > 0$
- We want to rule out the null
- Create a random variable X ranging over test sets
- And ask: how likely, if H_0 is true, is it that among these tests sets we would see the $\delta(x)$ we did see?
  - $P(\delta(X) \geq \delta(x) | H_0 \text{ true})$
  - This is the probability that we would see assuming the null
  - If H_0 is true but $\delta(x)$ is huge, this is surprising!
    - Reject the null!

- In NLP, we don't tend to use parametric tests like t-tests
- Instead, we use non-parametric tests based on sampling
  - artificially creating many versions of the setup
  - Two popular approaches: approximation randomization and bootstrap test

BOOTSTRAP TEST

- Can apply to any metric (accuracy, precision, recall, F1,…)
- **Bootstrap** means to repeatedly draw large numbers of smaller samples with replacement (called **bootstrap samples**) from an original larger sample
- This method only makes the assumption that the sample is representative of the population

- Consider a baby classification example with a test set x of 10 documents, using accuracy as metric

	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

- Now consider constructing bootstrap samples, say b=10,000 from test sets

	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
...													
$x^{(b)}$													

- Now we have a distribution
- So we just count how many times the $\delta(x')$ we found exceeds the expected 0 value by $\delta(x)$ or more
  - P-value: $\sum_i 1(\delta(x^{(i)}) - \delta(x) \geq 0)$

- It is slightly more complicated
- We didn't draw the bootstrap samples from a distribution with 0 mean
  - Biased by 0.2 in favor of A
- P-value: $\sum_i 1\left(\delta\left(x^{(i)}\right) - \delta(x) \geq \delta(x)\right)$

- Suppose
- We have 10,000 test sets $x(i)$ and a threshold of .01
- And in only 47 of the test sets do we find that $\delta\big(x(i)\big) \geq 2\delta(x)$
- The resulting p-value is .0047
- This is smaller than .01, indicating $\delta(x)$ is indeed sufficiently surprising
- And we reject the null hypothesis and conclude A is better than B.

After Berg-Kirkpatrick et al (2012)

function BOOTSTRAP(test set x, num of samples b) **returns** p-*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** $\delta(x^{(i)}) > 2\delta(x)$
p-value(x) $\approx \frac{s}{b}$ # on what % of the b samples did algorithm A beat expectations?
return p-value(x) # if very few did, our observed δ is probably not accidental