NLP: Classification

Dan Garrette
dhg@cs.utexas.edu

September 16, 2013

1 Classification Tasks

- Language identification: determine the language that a text is written in
- Spam filtering: label emails, tweets, blog comments as spam or not spam
- Routing: label emails to approprate people in an organization (complaints, tech support, order status, etc)
- Sentiment analysis: label some text as being positive or negative (polarity classification)

Example: Sentiment

- Determine the sentiment (positive vs. negative) of, for example, a tweet
- "Probably the worst movie of the century"
- One idea: Compare of "positive" words (good, great, best) to "negative" words (bad, terrible, worst).
  - Humans give insufficent lists. Learning a sufficent list is hard.
  - Words mean different things in different contexts:
    "This thing is *a great deal*. Definitely worth the money."
    "*A great deal* of media attention surrounded the event."
    "It's *a great deal*... if you're looking to looking to waste your money."
  - Some words can flip polarity:
    "It's **not** a **good** investment."
    "I thought it would be **terrible**, **but** I was so wrong."
  - Multi-word expressions:
    "The movie was shit."
    "The movie was the shit."
  - Subtlety:
    "If this movie's your thing, don't bother talking to me."
    "Great plot, great acting, great cast, but it just doesn't hold up."

  - Can depend on the target:
    "Unpredicatable plot"
    "Unpredicatable steering"

- Additional Features
  - Ngrams: "must buy", "couldn't care less"
  - Casing: uppercase words are often subjective
  - Punctuation: lots of ! or ? can indicate subjectivity
  - Emoticons: :) vs. :(

2 Rule-Based System

- Write prediction rules: "If contains X and Y but not Z, then 'positive'"
- What happens when multiple rules apply but conflict?
  - Order the rules according to their accuracy?
  - Assign weights to the rules?
- Problems
  - Time-consuming and expensive to write rules.
  - High precision, low recall
  - Rules have to be manually tailored to each dataset (unpredictable vs. unpredictable)
  - Expensive to update (new expressions, new slang, new abrvs)

3 Learning

- If we have examples, we can learn a function mapping texts to categories
- Often probabilistic
- Instead of rules, we use features
- Features are automatically weighted based on statistics in the training data.
- Features are dimensions in space.
- Learn a boundary between classes.
- Boundary used to classify new texts.

Some Data

```
start=B, end=ia, location
start=B, end=er, person
start=M, end=ia, person
start=L, end=ia, location
start=N, end=er, location
start=B, end=ia, location
start=E, end=nd, location
start=N, end=ia, location
start=A, end=er, person
start=L, end=ke, person
```

Our Goal

- Learn a function that maps features to the most likely label
- best_label = $\text{argmax}_{\text{label}}\ p(label \mid \text{features})$

Direct Posterior Parameter Estimation from Data

- $p(label \mid features) = \frac{C(instances\ with\ label\ and\ feature)}{C(instances\ with\ features)}$
  - $p(label = location \mid start = B, end = er) = \frac{0}{1} = 0.0$
  - $p(label = person \mid start = B, end = er) = \frac{1}{1} = 1.0$
  - $p(label = location \mid start = B, end = ia) = \frac{2}{2} = 1.0$
  - $p(label = person \mid start = B, end = ia) = \frac{0}{2} = 0.0$
- This doesn't work very well
- Sparsity: any particular feature combination is rare, hard to generalize
- Even worse when every word in a text is a feature
  - Every text would be unique, and there would be no generalization at all
  - Couldn't label new instances

Bayes Rule

- $p(label \mid features) = \frac{p(features|label) \cdot p(label)}{p(features)}$
- best_label = $\text{argmax}_{\text{label}}\ \frac{p(features|label) \cdot p(label)}{p(features)}$
- If labels = {A,B}: $\frac{p(features|A) \cdot p(A)}{p(features)}$ vs. $\frac{p(features|B) \cdot p(B)}{p(features)}$
- Denominator is always the same, so: $p(features \mid A) \cdot p(A)$ vs. $p(features \mid B) \cdot p(B)$
- Thus, to compute the **posterior**, we need two things
  - the likelihood of the evidence: $p(features \mid label)$

- the prior: $p(label)$

Direct Evidence Likelihood Estimation from Data

- $p(features \mid label) = \frac{C(instances\ with\ features\ and\ label)}{C(instances\ with\ label)}$
  - $p(start = B, end = er \mid label = location) = \frac{0}{6} = 0.0$
  - $p(start = B, end = er \mid label = person) = \frac{1}{4} = 0.25$
  - $p(start = B, end = ia \mid label = location) = \frac{2}{6} = 0.33$
  - $p(start = B, end = ia \mid p(label = person) = \frac{0}{4} = 0.0$
- Still problematic: still sparse, still lots of zeros, still hard to generalize

Naïve Bayes

- We want to disentangle the features for better generalization
- Compute each feature's probability independently
- Will be able to compute the probability of an instance from the features even if we haven't seen that particular combination of features before.
- Requires us to assume that features are independent
  - Not actually true! Language doesn't work like that.
  - But it's a simplifying assumption
  - "Naïve" assumption
- $p(features \mid label) = p(F_1, F_2, F_3, ... \mid label) = p(F_1 \mid label) \cdot p(F_2 \mid label) \cdot p(F_3 \mid label) \cdot ...$

Parameter Estimation from Data

- The prior
  - $p(label = location) = \frac{C(label=location)}{\sum_l C(label=l)} = \frac{6}{10} = 0.6$
  - $p(label = person) = \frac{C(label=person)}{\sum_l C(label=l)} = \frac{4}{10} = 0.4$
- Likelihood of the evidence
  - $p(start = B \mid label = location) = \frac{C(start=B,label=location)}{C(label=location)} = \frac{2}{6} = 0.33$
  - $p(start = B \mid label = person) = \frac{C(start=B,label=person)}{C(label=person)} = \frac{1}{4} = 0.25$
  - $p(end = ia \mid label = location) = \frac{C(end=ia,label=location)}{C(label=location)} = \frac{4}{6} = 0.67$
  - $p(end = ia \mid label = person) = \frac{C(end=ia,label=person)}{C(label=person)} = \frac{1}{4} = 0.25$

  - $p(end = nd \mid label = location) = \frac{C(end=nd, label=location)}{C(label=location)} = \frac{1}{6} = 0.17$
  - $p(end = nd \mid label = person) = \frac{C(end=nd, label=person)}{C(label=person)} = \frac{0}{4} = 0.0$

Naïve Probabilities

- Before
  - $p(start = B, end = ia \mid label = location) = \frac{2}{6} = 0.33$
  - $p(start = B, end = ia \mid p(label = person) = \frac{0}{4} = 0.0$
- Now
  - $p(start = B \mid label = location) \cdot p(end = ia \mid label = location) = 0.33 \cdot 0.67 = 0.22$
  - $p(start = B \mid label = location) \cdot p(end = ia \mid label = person) = 0.25 \cdot 0.25 = 0.06$

Classifying

- We get a **new** instance. Need to determine its label.
- Works even if we haven't seen the particular combination of features
- `start=L, end=er`
- $p(features \mid A) \cdot p(A)$ vs. $p(features \mid B) \cdot p(B)$
- $p(start = L \mid location) \cdot p(end = er \mid location) \cdot p(location) = \frac{1}{6} \cdot \frac{1}{6} \cdot \frac{6}{10} = 0.02$
- $p(start = L \mid person) \cdot p(end = er \mid person) \cdot p(person) = \frac{1}{4} \cdot \frac{2}{4} \cdot \frac{4}{10} = 0.06$
- So, it's more likely a *person*

Why Naïve Bayes?

- Modularity: separate out individual features and prior
- Helps us deal with sparsity
  - Particular feature combinations are rare
  - Individual features are less sparse
  - Still have some features that appear only with one label (meaning zero probaiblities), but this is less common
- Priors
  - Controls how much the base label distribution affects the probabilities
  - Can be automatically calculated from data (as we've seen)
  - Can be set from outside knowledge, if available
    - Imagine we are explicitly told that 3/4 of named entities are people
    - $p(label = person) = 0.75$

    - $p(start = L \mid location) \cdot p(end = er \mid location) \cdot p(location) = \frac{1}{6} \cdot \frac{1}{6} \cdot 0.25 = 0.007$
    - $p(start = L \mid person) \cdot p(end = er \mid person) \cdot p(person) = \frac{1}{4} \cdot \frac{2}{4} \cdot 0.75 = 0.09$
    - So the likelihood of *person* is even higher
  - Useful for injecting linguistic knowledge into a learned model

// Day 2

4 Smoothing

Classify *Austria*:

- The prior
  - $p(label = location) = \frac{C(label=location)}{\sum_l C(label=l)} = \frac{6}{10} = 0.6$
  - $p(label = person) = \frac{C(label=person)}{\sum_l C(label=l)} = \frac{4}{10} = 0.4$
- Likelihood of the evidence
  - $p(end = ia \mid label = location) = \frac{C(end=ia,label=location)}{C(label=location)} = \frac{4}{6} = 0.67$
  - $p(end = ia \mid label = person) = \frac{C(end=ia,label=person)}{C(label=person)} = \frac{1}{4} = 0.25$
  - $p(start = A \mid label = location) = \frac{C(start=A,label=location)}{C(label=location)} = \frac{0}{6} = 0.00$
  - $p(start = A \mid label = person) = \frac{C(start=A,label=person)}{C(label=person)} = \frac{1}{4} = 0.25$
- The posterior
  - $p(label = location \mid start = A, end = ia)$
    $= p(label = location) \cdot p(start = A \mid label = location) \cdot p(end = ia \mid label = location)$
    $= 0.6 \cdot 0.0 \cdot 0.67 = \mathbf{0.0}$
  - $p(label = person \mid start = A, end = ia)$
    $= p(label = person) \cdot p(start = A \mid label = person) \cdot p(end = ia \mid label = person)$
    $= 0.4 \cdot 0.25 \cdot 0.25 = \mathbf{0.025}$
- Since $0.025 > 0.0$, we pick the label *person*
- This seems wrong
  - $end = ia$ leans strongly toward *location* (0.67 vs. 0.25)
  - *location* is more likely *a priori* (0.6 vs 0.4)
  - While $start = A$ is leans toward *person*, it's not so strong (0.25 vs. 0.00)
  - Moreover, we have very little evidence regarding $start = A$, so we probably shouldn't give much weight to it during classification
- Why did we get the wrong answer?
  - Since we had no evidence for $label = location$ AND $start = A$, we got a zero count, which gave us a zero probability.

- So, basically, the $start = A$ feature ended up being the only thing that mattered.
- Even though $start = A$ should have been the least important thing in the calculation.

Add-λ smoothing

- We need to avoid having any zero counts.
- More generally, we need to deal with cases in which we have so few counts that it's hard to generalize
  - $C(A) = 1000, C(B) = 2$ wouldn't be considered a lack of evidence
  - $C(A) = 2, C(B) = 1$ probably doesn't have enough evidence to judge well
- One way to make all the numbers non-zero is to simply add 1 to all of them.
- So if the count was 0, it becomes 1, 1 becomes 2, 100 becomes 101.
- *start* values = {B,M,L,N,E,A}, *end* values = {ia,er,nd,ke}

(Feature, Label)	C(F,L)	+1	p(F|L)	p_{+1}(F|L)	(Feature, Label)	C(F,L)	+1	p(F|L)	p_{+1}(F|L)
(B, loc)	2	3	2/6=0.33	3/12=0.25	(B, per)	1	2	1/4=0.25	2/10=0.20
(M, loc)	0	1	0/6=0.00	1/12=0.08	(M, per)	1	2	1/4=0.25	2/10=0.20
(L, loc)	1	2	1/6=0.17	2/12=0.17	(L, per)	1	2	1/4=0.25	2/10=0.20
(N, loc)	2	3	2/6=0.33	3/12=0.25	(N, per)	0	1	0/4=0.00	1/10=0.10
(E, loc)	1	2	1/6=0.17	2/12=0.17	(E, per)	0	1	0/4=0.00	1/10=0.10
(A, loc)	0	1	0/6=0.00	1/12=0.08	(A, per)	1	2	1/4=0.25	2/10=0.20
(·, loc)	6	12			(·, per)	4	10		
(ia, loc)	4	5	4/6=0.67	5/10=0.50	(ia, per)	1	2	1/4=0.25	2/8=0.25
(er, loc)	1	2	1/6=0.17	2/10=0.20	(er, per)	2	3	2/4=0.50	3/8=0.38
(nd, loc)	1	2	1/6=0.17	2/10=0.20	(nd, per)	0	1	0/4=0.00	1/8=0.13
(ke, loc)	0	1	0/6=0.00	1/10=0.10	(ke, per)	1	2	1/4=0.25	2/8=0.25
(·, loc)	6	10			(·, per)	4	8		

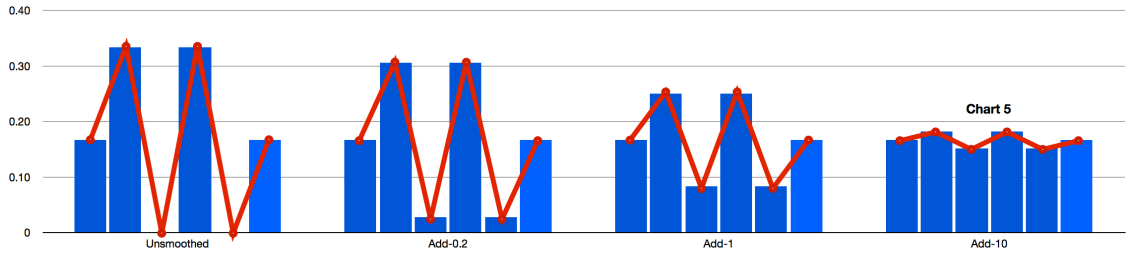
- The count of each (Feature, Label) pair goes up by 1, but the total count for the label across all features goes up by $(1 \cdot (\#\ of\ Feature\ Values))$
  - Since we add 1 additional count for each (F,L) — even the ones we've never seen before — and there is one (F,L) pair for each possible F, there are $1 \cdot |Feature\ Values|$ additional counts for the label.
  - $\sum_{f \in Features}(C(f, L) + 1) = \left(\sum_{f \in Feature\ Values} C(f, L)\right) + |Feature\ Values|$
  - So now, $p(feature \mid label) = \frac{C(feature,label)+1}{C(label)+(1 \cdot |Feature\ Values|)}$
  - If we didn't increase the denominator:

$$p(start = B \mid label = location) = \frac{C(start=B,label=location)+1}{C(label=location)} = \frac{2+1}{6} = \frac{3}{6} = 0.50$$
$$p(start = M \mid label = location) = \frac{C(start=M,label=location)+1}{C(label=location)} = \frac{0+1}{6} = \frac{1}{6} = 0.17$$

$$p(start = L \mid label = location) = \frac{C(start{=}L, label{=}location)+1}{C(label{=}location)} = \frac{1+1}{6} = \frac{2}{6} = 0.33$$
$$p(start = N \mid label = location) = \frac{C(start{=}N, label{=}location)+1}{C(label{=}location)} = \frac{2+1}{6} = \frac{3}{6} = 0.50$$
$$p(start = E \mid label = location) = \frac{C(start{=}E, label{=}location)+1}{C(label{=}location)} = \frac{1+1}{6} = \frac{2}{6} = 0.33$$
$$p(start = A \mid label = location) = \frac{C(start{=}A, label{=}location)+1}{C(label{=}location)} = \frac{0+1}{6} = \frac{1}{6} = 0.17$$

  - But $0.50 \cdot 0.17 \cdot 0.33 \cdot 0.50 \cdot 0.33 \cdot 0.17 = 2.0$
  - Since all the values in a probaiblity distribution must sum to 1.0, we no longer have a probability distribution
  - Adding the correct amount to the denominator fixes this problem.

- Smoothing flattens out the distribution



  - (Partial) counts are "stolen" from high-count items and "given" to low-count items
  - Extreme values are smoothed the most
  - The impact of smoothing is depends on the relative values of λ and the counts.
  - If there are few overall counts, then a relatively large λ will over-smooth, making all values approximately the same likelihood.
  - If there are many overall counts, then a relatively small λ will not make much impact.
  - $\lambda = 0$ means no smoothing
  - $\lambda = \infty$ means uniform distributiom (all have equal probability)

- Smoothing changes the relative probabilities of things most when there are low overall counts
  - This is good because low counts means low evidence and, thus, low confidence
  - We want the impact of evidence to be strong only when we have high confidence
  - For naïve Bayes, we always compare $p(F \mid L_1)$ to $p(F \mid L_2)$
  - If counts $C(F, L_1)$ and $C(F, L_2)$ are small, we want smoothing to push the relative probabilities together, thus giving them a smaller impact in the NB calculation. (The most extreme example is equal probabilities, which contribute no information to NB vs zero counts, which completely dominate the NB calculation).
  - If counts $C(F, L_1)$ and $C(F, L_2)$ are large, we do not want smoothing to have a large impact on the relative probabilities

  - For things with low counts (0 vs. 1), smoothing pushes the relative probabilities close together
    If $C(A) = 1$, $C(B) = 0$
    - $p(A) = \frac{1}{1} = \mathbf{1.0}$ and $p(B) = \mathbf{0.0}$

    * $p_{+1}(A) = \frac{2}{3} = \mathbf{0.67}$ and $p_{+1}(B) = \mathbf{0.33}$

  If $C(A) = 2$, $C(B) = 1$

    * $p(A) = \frac{2}{3} = \mathbf{0.67}$ and $p(B) = \mathbf{0.33}$
    * $p_{+1}(A) = \frac{3}{5} = \mathbf{0.60}$ and $p_{+1}(B) = \mathbf{0.40}$

  – For things with non-trivial counts, the smoothed relative probabilities don't change much. If $C(A) = 30$, $C(B) = 0$

    * $p(A) = \frac{30}{30} = \mathbf{1.0}$ and $p(B) = \mathbf{0.0}$
    * $p_{+1}(A) = \frac{31}{32} = \mathbf{0.97}$ and $p_{+1}(B) = \mathbf{0.03}$

  If $C(A) = 20$, $C(B) = 10$

    * $p(A) = \frac{20}{30} = \mathbf{0.67}$ and $p(B) = \mathbf{0.33}$
    * $p_{+1}(A) = \frac{21}{32} = \mathbf{0.66}$ and $p_{+1}(B) = \mathbf{0.44}$

Classify *Austria* with Add-1 smoothing

- The prior
  - We do *not* need (or want) to smooth the prior distribution $p(Label)$
  - Smoothing is useful when we have very low or zero counts
  - Counts of labels will never be so low that we can't estimate them
- Likelihood of the evidence
  - $p(end = ia \mid label = location) = \frac{C(end=ia,label=location)+1}{C(label=location)+|Feature\ Values|} = \frac{4+1}{6+4} = \frac{5}{10} = 0.50$
  - $p(end = ia \mid label = person) = \frac{C(end=ia,label=person)}{C(label=person))+|Feature\ Values|} = \frac{1+1}{4+4} = \frac{2}{8} = 0.25$
  - $p(start = A \mid label = location) = \frac{C(start=A,label=location)}{C(label=location))+|Feature\ Values|} = \frac{0+1}{6+6} = \frac{1}{12} = 0.08$
  - $p(start = A \mid label = person) = \frac{C(start=A,label=person)}{C(label=person))+|Feature\ Values|} = \frac{1+1}{4+6} = \frac{2}{10} = 0.20$
  - No longer have zeros (this is essential)
  - $start = A$ relative probabilities are pushed closer together than $end = ia$ since there is less evidence for either label
- The posterior
  - $p(label = location \mid start = A, end = ia)$
    $= p(label = location) \cdot p(start = A \mid label = location) \cdot p(end = ia \mid label = location)$
    $= 0.6 \cdot 0.08 \cdot 0.50 = \mathbf{0.024}$
  - $p(label = person \mid start = A, end = ia)$
    $= p(label = person) \cdot p(start = A \mid label = person) \cdot p(end = ia \mid label = person)$
    $= 0.4 \cdot 0.20 \cdot 0.25 = \mathbf{0.020}$
- Since $0.024 > 0.020$, we pick the label *location*
- So smoothing fixed this classification example!

5 Features occurring more than once per instance

- Up to this point we have only worked with instances that have exactly one value of each feature per instance.
- But sometimes we want to classify data that has feaures that appear multiple times per instance, possibly with the same values.
- A canoncial example of this is when we are classifying text and we simply use the words in the text as features.
- The dataset:

```
the movie was great and the acting was great , so i loved it, positive
i loved the movie dispite the terrible acting, positive
the plot was great , but overall it was terrible, negative
that movie was terrible, negative
the movie had a terrible ending, but was great anyway, positive
what a terrible movie, negative
i've never seen anything so terrible, negative
```

- Could have stopwords removed and be transformed to:

```
word=movie,word=great,word=acting,word=great,word=loved,5words=true,positive
word=loved,word=movie,word=terrible,word=acting,5words=true,positive
word=plot,word=great,word=terrible,5words=true,negative
word=movie,word=terrible,5words=false,negative
word=movie,word=terrible,word=ending,word=great,5words=true,positive
word=terrible,word=movie,5words=false,negative
word=terrible,5words=true,negative
```

- 2 features: *word* and *5words*

Counting with duplicate features

- Since there is no longer one feature of each time per instance, we have to make our probability calculations more precise.
- $p(feature = F \mid label = L) = \frac{C(feature=F,label=L)}{\sum_f C(feature=f,label=L)}$
- So we are no longer dividing the number of instance with the label and feature by the number of instances with the label.
- Instead, we have to count the number of (label,value) pairs extracted from the data for the particular feature, including multiple identical pairs in the same instance
- The probability $p(feature = F \mid label = L)$ is the proportion of times the particular feature was F in an instance with label L out of the total number of times the particular feature appeared in an instance with label L

- $p(word = great \mid label = positive) = \frac{C(word=great,label=positive)}{\sum_f C(word=f,label=positive)} = \frac{3}{13} = 0.23$
  $p(word = great \mid label = negative) = \frac{C(word=great,label=negative)}{\sum_f C(word=f,label=negative)} = \frac{1}{8} = 0.13$
- If we only used the count of instances as the denominator, we would not end up with a probability distribution.
- Note that we still calculate the prior the same way: percentage of *instances* that have the particular label.

6 Evaluating

- How do we tell if our classifier works?
  - Evaluating on the training data doesn't tell us about performance on new data
- How do we set hyper-parameters (e.g. λ) without cheating?
  - Tuning them to the test set wouldn't tell us about performance on new data

Data Split

- Divide labeled data into three groups: **train**, **development (dev)**, and **test**
- Use only the *train* data to set parameters (take counts)
- Test against the *dev* data using various choices for hyper-parameters (e.g. λ) to figure out the best settings.
- When all development is done, and you are ready to get final results for publication, evaluate *just once* on the *test* data, and record the results as your final results.

Data Size

- Larger training set means that we will have better count estimates and, therefore, better model parameter estimates
- Larger dev and test sets mean that we can trust the results more

Accuracy

- Proportion of test instances that are labeled correctly: $\frac{C(correct)}{C(total)}$
- 100 instances, 70 correctly labeled: 70% correct
- Simple and intuitive, but can be misleading
- 100 instances, 80 are *not spam*
  - If the system just always outputs *not spam*, then accuracy is 80%

- But the model is worthless!

Precision and Recall

- Precision: Out of those that we labeled "spam", what proportion are actually spam?
  $p(is\ spam \mid says\ \text{“}spam\text{”})$
- Recall: Out of those that are actually spam, what proportion did we correctly label as "spam"?
  $p(says\ \text{“}spam\text{”} \mid is\ \text{“}spam\text{”})$
- Categorize kinds of correctness and mistakes for *spam*
  true positive (tp): model says "spam", actually spam
  false positive (fp): model says "spam", actually not spam
  false negative (fn): model says "not spam", actually spam
  true negative (tn): model says "not spam", actually not spam
- $precision = \frac{tp}{tp+fp}$ $\quad$ $recall = \frac{tp}{tp+fn}$
- F_1-score: A way of combining precision and recall into one number (using the harmonic mean)
  - $F_1 = 2 \cdot \frac{P \cdot R}{P+R}$
- 100 instances:
  - P & R on *spam*

		model says	
		spam	not spam
truth	spam	TP = 20	FN = 10
	not spam	FP = 15	TN = 55

    * Accuracy is 0.75
    * Precision $= \frac{tp}{tp+fp} = \frac{20}{20+15} = 0.57$
    * Recall $= \frac{tp}{tp+fn} = \frac{20}{20+10} = 0.67$
    * $F_1 = 2 \cdot \frac{P \cdot R}{P+R} = 2 \cdot \frac{0.57 \cdot 0.67}{0.57+0.67} = 0.62$

  - P & R on *not spam*

		model says	
		spam	not spam
truth	spam	TN = 20	FP = 10
	not spam	FN = 15	TP = 55

    * Accuracy is 0.75
    * Precision $= \frac{tp}{tp+fp} = \frac{55}{55+10} = 0.85$
    * Recall $= \frac{tp}{tp+fn} = \frac{55}{55+15} = 0.79$
    * $F_1 = 2 \cdot \frac{P \cdot R}{P+R} = 2 \cdot \frac{0.85 \cdot 0.79}{0.85+0.79} = 0.89$
- 100 instances, 80 are *not spam*, and the system just always outputs *not spam*

– P & R on *spam*

		model says	
		spam	not spam
truth	spam	TP = 0	FN = 20
	not spam	FP = 0	TN = 80

  * Accuracy is 0.80
  * Precision = $\frac{tp}{tp+fp} = \frac{0}{0+0} = NaN$
  * Recall = $\frac{tp}{tp+fn} = \frac{0}{0+20} = 0.0$
  * $F_1 = 2 \cdot \frac{P \cdot R}{P+R} = ?$

– P & R on *not spam*

		model says	
		spam	not spam
truth	spam	TN = 0	FP = 20
	not spam	FN = 0	TP = 80

  * Accuracy is 0.80
  * Precision = $\frac{tp}{tp+fp} = \frac{80}{80+20} = 0.80$
  * Recall = $\frac{tp}{tp+fn} = \frac{80}{80+0} = 1.00$
  * $F_1 = 2 \cdot \frac{P \cdot R}{P+R} = 2 \cdot \frac{0.80 \cdot 1.00}{0.80+1.00} = 0.89$

7 Semi-Supervised Learning for Naïve Bayes

- At this point, we know how to train a naïve Bayes classifier from fully-supervised data (every training instance has a label)
- Can we use (lots of) unlabeled data to improve a NB classifier trained on just a small amount of labeled data?
- Can we learn a NB classifier with out any labeled data?

An excellent tutorial that carefully explains everything necessary to implement Gibbs sampling for naïve Bayes:

- Philip Resnik and Eric Hardisty. "Gibbs Sampling for the Uninitiated". June 2010. `http://www.cs.umd.edu/~hardisty/papers/gsfu.pdf`

Partial Supervision: The Idea

- We know how to use NB to classify if we have labeled data
- If we have unlabeled data, we can assign labels to all the instances, then it because labeled data, and we can use it for NB
- We don't want to just assign labels and assume that they are correct

- So we will make initial guesses for those labels, and iteratively revise them
- Critically, we won't revise by simply picking the most likely label. We will choose probabilistically, based on the relatively probaiblities of the different labels.

The Algorithm

1. Make a NB classifier from any labeled data available.
2. For each unlabeled instance:
    a. Use this classifier to calculate the probabilities of each label.
    b. Make a probabilty distribution out of all label probabilties (normalize them so that they add up to 1.0)
    c. Sample from the probability distribution by choosing a label with likelihood according to its probability
    d. Assign that chosen label as the label for the instance
3. For some large number of iterations, for each unlabeled instance:
    a. Make a NB classifier from all the instances and their current labels.
    c. Use this classifier to calculate the probabilities of each label.
    c. Make a probabilty distribution out of all label probabilties (normalize them so that they add up to 1.0)
    d. Sample from the probability distribution by choosing a label with likelihood according to its probability
    e. Assign that chosen label as the revised label for the instance
    f. Copy the instance with its new label into a "repository" of labeled instances.
4. Make a final NB classifier from all of the labeled instances in the "repository" that we built up while we were sampling.

- We don't revise the labels of the supervised instances because we assume that they are correct (unless we don't trust them, or think they are out-of-date).
- Always making probabilistic label choices prevents us from just accepting the MLE and getting stuck.
- The choices will allow us to "walk" around the space of labels, where most of the walking happens in area that lead to better parameters.

Why does this work?

- Basically, it learns how group associated feature values together, and how to push different labels apart.
- We start with a "best guess" based on information we have, and revise our guesses over time.
- We don't just walk "uphill" based on the MLE.

- Instead we allow ourselves to explore lots of label choices, some of which may be very different from the MLE.
- As we become more confident in a judgement, its label is less likely to change because more of the similar instances will be using that label, which gives it a high probability.

Markov Chain Monte Carlo

- Markov: Uses the "Markov principle" which says to make decisions based only on recent information
  - We use only the current set of labels, not the full history of labels.
- Chain: Start with an initial set of labels, use it to change one label, then move to the next label and change it based on the new set of labels. Repeat for a long time.
- Monte Carlo: Each step in the chain has chance involved
  - We choose the next label by chance according to the relative probabilities of the labels.
  - Named for Monte Carlo in Monaco and the Monte Carlo Casino

8 Clustering

- Until now, all our classifiers have been learned from fully-supervised data (every training instance has a label)
- Can we learn a classifier when we have no labels at all?
- **Clustering** is when we try to automatically induce groupings without labels.

K-Means

1. Plot all instances in space
2. Choose K of them to be initial centroids
3. For each instance, associate it with its nearest centroid
4. Update each centroid to the centroid of its currently-assocated instances
5. Repeat step 3-4 until the centroids stop moving
6. For new instances, classify based on their nearest centroid