RECAP: THE COURSE SO FAR…
First Lecture - Review

• Admin stuff
• Review – Why to scale, how to count and what to count
  • How: O(…)

First Lecture – NOT Review

• Admin stuff – new!
• William’s office hours: 11am Mon
• And …

You have the option of replacing one assignment with an "open-ended extension"

- Roughly, this would be an extension to an existing assignment that would increase the programming effort by at least 50%.
- Your deliverable is a handout for the extension, like a TA would produce, and a solution key, including code.
- Example: for HW1B, implement the Rocchio algorithm as well as naive Bayes and compare them.
Why to scale: c. 2001 (Banko & Brill, ACL 2001)

Task: distinguish pairs of easily-confused words ("affect" vs "effect") in context
# What to count

<table>
<thead>
<tr>
<th>Operation</th>
<th>~ Time</th>
<th>x/100ns</th>
<th>x/10M ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>random access, RAM</td>
<td>100 ns</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>read 1 Mb sequentially - RAM</td>
<td>250,000 ns</td>
<td>2,500</td>
<td></td>
</tr>
<tr>
<td>random access, disk (seek)</td>
<td>10,000,000 ns</td>
<td>100,000</td>
<td>1</td>
</tr>
<tr>
<td>read 1Mb sequentially - net</td>
<td>10,000,000 ns</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>read 1Mb sequentially - disk</td>
<td>30,000,000 ns</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
What to count

• Compilers don’t warn Jeff Dean. Jeff Dean warns compilers.
• ....
• Memory access/instructions are qualitatively different from disk access
• Seeks are qualitatively different from sequential reads on disk
• Cache, disk fetches, etc work best when you stream through data sequentially
• Best case for data processing: stream through the data once in sequential order, as it’s found on disk.
First lecture: review

• Admin stuff
• Review – **Why** to scale, **how** to count and **what** to count

• What sort of computations do we want to *do* in (large-scale) machine learning programs?
  – *Probability*
PROBABILITY AND SCALABILITY: LEARNING AND COUNTING
Task: distinguish pairs of easily-confused words ("affect" vs "effect") in context
Why More Data Helps

• Data:
  – All 5-grams that appear >= 40 times in a corpus of 1M English books
    • approx 80B words
    • 5-grams: 30Gb compressed, 250-300Gb uncompressed
    • Each 5-gram contains frequency distribution over years
  – Wrote code to compute
    • Pr(A,B,C,D,E | C=affect or C=effect)
    • Pr(any subset of A,…,E | any other fixed values of A,…,E with C=affect V effect)

• Observations [from playing with data]:
  – Mostly effect not affect
  – Most common word before affect is not
  – After not effect most common word is a
  – …
Recipe for making a joint distribution of $M$ variables:

1. Make a truth table listing all combinations of values of your variables (if there are $M$ Boolean variables then the table will have $2^M$ rows).
2. For each combination of values, say how probable it is.
3. If you subscribe to the axioms of probability, those numbers must sum to 1.

Example: Boolean variables $A$, $B$, $C$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.30</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.05</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.10</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.05</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.05</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.10</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.25</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.10</td>
</tr>
</tbody>
</table>
Some of the Joint Distribution

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>is</td>
<td>the</td>
<td>effect</td>
<td>of</td>
<td>the</td>
<td></td>
<td>0.00036</td>
</tr>
<tr>
<td>is</td>
<td>the</td>
<td>effect</td>
<td>of</td>
<td>a</td>
<td></td>
<td>0.00034</td>
</tr>
<tr>
<td>.</td>
<td>The</td>
<td>effect</td>
<td>of</td>
<td>this</td>
<td></td>
<td>0.00034</td>
</tr>
<tr>
<td>to</td>
<td>this</td>
<td>effect</td>
<td>:</td>
<td>“</td>
<td></td>
<td>0.00034</td>
</tr>
<tr>
<td>be</td>
<td>the</td>
<td>effect</td>
<td>of</td>
<td>the</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>not</td>
<td>the</td>
<td>effect</td>
<td>of</td>
<td>any</td>
<td></td>
<td>0.00024</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>does</td>
<td>not</td>
<td>affect</td>
<td>the</td>
<td>general</td>
<td></td>
<td>0.00020</td>
</tr>
<tr>
<td>does</td>
<td>not</td>
<td>affect</td>
<td>the</td>
<td>question</td>
<td></td>
<td>0.00020</td>
</tr>
<tr>
<td>any</td>
<td>manner</td>
<td>affect</td>
<td>the</td>
<td>principle</td>
<td></td>
<td>0.00018</td>
</tr>
</tbody>
</table>
An experiment: how useful is the brute-force joint classifier?

• Extracted all affect/effect 5-grams from an old Reuters corpus
  – about 20k documents
  – about 723 n-grams, 661 distinct
  – Financial news, not novels or textbooks

• Tried to predict center word with:
  – \( \Pr(C \mid A=a, B=b, D=d, E=e) \)
  – then \( \Pr(C \mid A, B, D) \)
  – then \( \Pr(C \mid B, D) \)
  – then \( \Pr(C \mid B) \)
  – then \( \Pr(C) \)
EXAMPLES

- “The cumulative _ of the” $\rightarrow$ effect (1.0)
- “Go into _ on January” $\rightarrow$ effect (1.0)
- “From cumulative _ of accounting” not present in train data
  - Nor is “From cumulative _ of _”
  - But “_ cumulative _ of _” $\rightarrow$ effect (1.0)
- “Would not _ Finance Minister” not present
  - But “_ not _ _ _” $\rightarrow$ affect (0.9625)
## Performance summary

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Used</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(C</td>
<td>A,B,D,E)</td>
<td>101</td>
</tr>
<tr>
<td>P(C</td>
<td>A,B,D)</td>
<td>157</td>
</tr>
<tr>
<td>P(C</td>
<td>B,D)</td>
<td>163</td>
</tr>
<tr>
<td>P(C</td>
<td>B)</td>
<td>244</td>
</tr>
<tr>
<td>P(C)</td>
<td>58</td>
<td>31</td>
</tr>
</tbody>
</table>

Is this a useful density estimate?
What Have We Learned?

• Counting’s not enough -?
• Counting goes a long way with big data -?

• Big data can sometimes be made small
  – For a specific task, like this one
  – It’s all in the data preparation -?

• Often density estimation is more important than classification

• Counts are a good? density estimator
Density Estimation

• Our Joint Distribution learner is our first example of something called **Density Estimation**

• A Density Estimator learns a mapping from a set of attributes values to a Probability

```
  Input Attributes --> Density Estimator --> Probability
```
Density Estimation

• Compare it against the two other major kinds of models:

  - **Classifier**
    - Input Attributes
    - Prediction of categorical output or class
      - One of a few discrete values
  - **Density Estimator**
    - Input Attributes
    - Probability
  - **Regressor**
    - Input Attributes
    - Prediction of real-valued output
Density Estimation ➔ Classification

To classify $x$
1. Use your estimator to compute $\hat{P}(x,y_1), \ldots, \hat{P}(x,y_k)$
2. Return the class $y^*$ with the highest predicted probability

Ideally is correct with $P(x,y^*) = \frac{\hat{P}(x,y^*)}{\hat{P}(x,y_1) + \ldots + \hat{P}(x,y_k)}$

Binary case: predict POS if $\hat{P}(x) > 0.5$
Classification vs Density Estimation

Classification

Density Estimation
Classification vs density estimation
PROBABILITY AND SCALABILITY: NAÏVE BAYES

Second most scalable learning method in the world?
Performance …

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Used</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(C</td>
<td>A,B,D,E)</td>
<td>101</td>
</tr>
<tr>
<td>P(C</td>
<td>A,B,D)</td>
<td>157</td>
</tr>
<tr>
<td>P(C</td>
<td>B,D)</td>
<td>163</td>
</tr>
<tr>
<td>P(C</td>
<td>B)</td>
<td>244</td>
</tr>
<tr>
<td>P(C)</td>
<td>58</td>
<td>31</td>
</tr>
</tbody>
</table>

• Is this good performance?
• If we care about recall, what should we do?
Naïve Density Estimation

What’s an alternative to the joint distribution?

The naïve model generalizes strongly:

Assume that each attribute is distributed independently of any of the other attributes.
Using the Naïve Distribution

• Once you have a Naïve Distribution you can easily compute any row of the joint distribution.
• Suppose $A$, $B$, $C$ and $D$ are independently distributed. What is $P(A \land \neg B \land C \land \neg D)$?
Using the Naïve Distribution

• Once you have a Naïve Distribution you can easily compute any row of the joint distribution.
• Suppose A, B, C and D are independently distributed. What is \( P(A ^ \sim B ^ C ^ \sim D) \)?

\[ P(A) \, P(\sim B) \, P(C) \, P(\sim D) \]
Naïve Distribution General Case

• Suppose $X_1, X_2, \ldots, X_d$ are independently distributed.

\[
Pr(X_1 = x_1, \ldots, X_d = x_d) = Pr(X_1 = x_1) \cdot \ldots \cdot Pr(X_d = x_d)
\]

• So if we have a Naïve Distribution we can construct any row of the implied Joint Distribution on demand.

• How do we learn this?
Learning a Naïve Density Estimator

\[ P(X_i = x_i) = \frac{\text{# records with } X_i = x_i}{\text{# records}} \quad \text{MLE} \]

\[ P(X_i = x_i) = \frac{\text{# records with } X_i = x_i + mq}{\text{# records} + m} \quad \text{Dirichlet (MAP)} \]

Another trivial learning algorithm!
Is this an interesting learning algorithm?  No

• For n-grams, what is $\hat{P}(C=\text{effect} \mid A=\text{will})$?
• In joint: $\hat{P}(C=\text{effect} \mid A=\text{will}) = 0.38$
• In naïve: $\hat{P}(C=\text{effect} \mid A=\text{will}) = \hat{P}(C=\text{effect}) = \frac{\#C=\text{effect}}{\#\text{totalNgrams}} = 0.94 (!)$

• What is $\hat{P}(C=\text{effect} \mid B=\text{no})$?
• In joint: $\hat{P}(C=\text{effect} \mid B=\text{no}) = 0.999$
• In naïve: $\hat{P}(C=\text{effect} \mid B=\text{no}) = \hat{P}(C=\text{effect}) = 0.94$
Can we make it interesting? Yes!

- Key ideas:
  - Pick the class variable $Y$
  - Instead of estimating $P(X_1,\ldots,X_n,Y) = P(X_1)\times\ldots\times P(X_n)\times P(Y)$, estimate $P(X_1,\ldots,X_n|Y) = P(X_1|Y)\times\ldots\times P(X_n|Y)$
  - Or, assume $P(X_i|Y) = Pr(X_i|X_1,\ldots,X_{i-1},X_{i+1},\ldots,X_n,Y)$
  - Or, that $X_i$ is conditionally independent of every $X_j$, $j\neq i$, given $Y$.

  - How to estimate?
    - use MAP or MLE for each $P(X_i=x|Y=y)$
The Naïve Bayes classifier – v1

- Dataset: each example has
  - A unique id \( id \)
    - Why? For debugging the feature extractor
  - One class label \( Y \) in \( \text{dom}(Y) \)
  - \( d \) attributes \( X_1,\ldots,X_d \)
    - Each \( X_i \) takes a discrete value in \( \text{dom}(X_i) \)
- You have a \textit{train} dataset and a \textit{test} dataset
- Assume:
  - the dataset doesn’t fit in memory
  - the model does
The Naïve Bayes classifier – v0

• You have a train dataset and a test dataset
• Initialize an “event counter” (hashtable) C
• For each example id, y, x₁,…..,xₙ in train:
  – C("Y=ANY") ++;  C("Y=y") ++
  – For j in 1..d:
    • C("Y=y ^ Xⱼ=xⱼ") ++
• For each example id, y, x₁,…..,xₙ in test:
  – For each y’ in dom(Y):
    • Compute Pr(y’,x₁,…..,xₙ) = \left( \prod_{j=1}^{d} \Pr(X_j = x_j | Y = y') \right) \Pr(Y = y')
  = \left( \prod_{j=1}^{d} \frac{\Pr(X_j=x_j,Y=y')}{\Pr(Y=y')} \right) \Pr(Y = y')
  – Return the best y’
The Naïve Bayes classifier – v0

• You have a *train* dataset and a *test* dataset
• Initialize an “event counter” ( hashtable) C
• For each example \(id, y, x_1, \ldots, x_d\) in train:
  – \(C(“Y=ANY”)++;\) \(C(“Y=y”)++\)
  – For \(j\) in 1..\(d\):
    • \(C(“Y=y \land X_j=x_j”)++\)
• For each example \(id, y, x_1, \ldots, x_d\) in test:
  – For each \(y’\) in \(\text{dom}(Y)\):
    • Compute \(\Pr(y’, x_1, \ldots, x_d) = \left( \prod_{j=1}^{d} \Pr(X_j = x_j | Y = y’) \right) \Pr(Y = y’)

\[
= \left( \prod_{j=1}^{d} \frac{C(X_j=x_j \land Y=y’)}{C(Y=y’)} \right) \frac{C(Y=y’)}{C(Y=ANY)}
\]
  This may overfit, so …

  – Return the best \(y’\)
The Naïve Bayes classifier – v1

• You have a *train* dataset and a *test* dataset
• Initialize an “event counter” (hashtable) C
• For each example *id*, *y*, *x₁*,…..,*xₙ* in *train*:
  – C("*Y=*ANY") ++;   C("*Y=*y") ++
  – For *j* in 1..*d*:
    • C("*Y=*y ∧ *X*=*x*j") ++
• For each example *id*, *y*, *x₁*,…..,*xₙ* in *test*:
  – For each *y'* in dom(*Y*):
    • Compute Pr(*y',*x₁*,…..,*xₙ*) = \[ \left( \prod_{j=1}^{d} \frac{\text{C}(\text{X}_j=x_j ∧ \text{Y}=\text{*y'}}{\text{C}(\text{Y}=\text{*y'}}+m \right) \frac{\text{C}(\text{Y}=\text{*y'})+m\text{q}_y}{\text{C}(\text{Y}=\text{ANY})+m} \]

\[ = \left( \prod_{j=1}^{d} \frac{\text{C}(\text{X}_j=x_j ∧ \text{Y}=\text{*y'})+m\text{q}_j}{\text{C}(\text{Y}=\text{*y'})+m} \right) \frac{\text{C}(\text{Y}=\text{*y'})+m\text{q}_y}{\text{C}(\text{Y}=\text{ANY})+m} \]

  – Return the best *y'*

This may underflow, so …
The Naïve Bayes classifier – v1

- You have a train dataset and a test dataset
- Initialize an “event counter” (hashtable) C
- For each example id, y, x₁,….xₙ in train:
  - C(“Y=ANY”) ++; C(“Y=y”) ++
  - For j in 1...d:
    - C(“Y=y ∧ X_j=x_j”) ++
- For each example id, y, x₁,….xₙ in test:
  - For each y’ in dom(Y):
    - Compute log Pr(y’,x₁,…..,xₙ) =
      \[
      \left( \sum_{j=1}^{d} \log \frac{C(X_j = x_j ∧ Y = y')}{{C(Y = y')} + m} \right) + \log \frac{C(Y = y')}{{C(Y = ANY)} + m}
      \]
    - Return the best y'

where:

- \( q_j = 1/|\text{dom}(X_j)| \)
- \( q_y = 1/|\text{dom}(Y)| \)
- \( m=1 \)
The Naïve Bayes classifier – v2

• For text documents, what features do you use?
• One common choice:
  – \( X_1 \) = first word in the document
  – \( X_2 \) = second word in the document
  – \( X_3 \) = third …
  – \( X_4 \) = …
  – …

• But: \( \Pr(X_{13} = \text{hockey} \mid Y = \text{sports}) \) is probably not that different from \( \Pr(X_{11} = \text{hockey} \mid Y = \text{sports}) \)…so instead of treating them as different variables, treat them as different copies of the same variable
The Naïve Bayes classifier – v2

- You have a *train* dataset and a *test* dataset
- Initialize an “event counter” (hashtable) C
- For each example *id, y, x₁,…..,xₖ* in *train*:
  - C(“Y=ANY”) ++;  C(“Y=y”) ++
  - For *j* in 1..d:
    - C(“Y=y ^ Xᵢ=xᵢ”) ++
- For each example *id, y, x₁,…..,xₖ* in *test*:
  - For each *y'* in dom(Y):
    - Compute Pr(y’,x₁,…..,xₖ) = \( \prod_{j=1}^{d} \frac{\Pr(X_j = x_j | Y = y')}{\Pr(Y = y')} \Pr(Y = y') \)
    - = \( \left( \prod_{j=1}^{d} \frac{\Pr(X_j = x_j, Y = y')}{\Pr(Y = y')} \right) \Pr(Y = y') \)
  - Return the best *y'*
The Naïve Bayes classifier – v2

• You have a train dataset and a test dataset
• Initialize an “event counter” (hashtable) C
• For each example id, y, x₁,…..,xₙ in train:
  – C(“Y=ANY”) ++; C(“Y=y”) ++
  – For j in 1..d:
    • C(“Y=y ^ X_j=x_j”) ++
• For each example id, y, x₁,…..,xₙ in test:
  – For each y’ in dom(Y):
    • Compute Pr(y’,x₁,…..,xₙ) = \left( \prod_{j=1}^{d} \frac{Pr(X_j = x_j | Y = y')} {Pr(Y = y')} \right) \frac{C(Y=y') + m q_y} {C(Y=ANY) + m}

= \left( \prod_{j=1}^{d} \frac{C(X_j=x_j \land Y=y') + m q_j} {C(Y=y') + m} \right) \frac{C(Y=y') + m q_y} {C(Y=ANY) + m}

  – Return the best y'}
The Naïve Bayes classifier – v2

- You have a *train* dataset and a *test* dataset
- Initialize an “event counter” (hashtable) \( C \)
- For each example \( id, y, x_1, \ldots, x_d \) in *train*:
  - \( C(“Y=\text{ANY}”) ++; \ C(“Y=\text{y}”) ++ \)
  - For \( j \) in 1..\( d_{id} \):
    - \( C(“Y=\text{y} \land X_j=\text{ANY})++; \ C(“Y=\text{y} \land X_j=x_j”) ++ \)
- For each example \( id, y, x_1, \ldots, x_d \) in *test*:
  - For each \( y’ \) in \( \text{dom}(Y) \):
    - Compute \( \Pr(y’,x_1,\ldots,x_d) = \left( \prod_{j=1}^{d} \Pr(X_i=x_j|Y=y’) \right) \Pr(Y=y’) \)
  
  \[
  = \left( \prod_{j=1}^{d} \frac{C(X_j=x_j \land Y=y’)+mq_j}{C(X_j=\text{ANY} \land Y=y’)+m} \right) \frac{C(Y=y’)+mq_y}{C(Y=\text{ANY})+m} 
  \]

  - Return the best \( y’ \)
The Naïve Bayes classifier – v2

• You have a train dataset and a test dataset
• Initialize an “event counter” (hashtable) C
• For each example id, y, x₁,…..,xₜ in train:
  – C(“Y=ANY”) ++;  C(“Y=y”) ++
  – For j in 1..dₜ:
    • C(“Y=y ^ X=ANY”)++;  C(“Y=y ^ Xₖ=xₖ”) ++
• For each example id, y, x₁,…..,xₜ in test:
  – For each y’ in dom(Y):
    • Compute Pr(y’,x₁,…..,xₜ) = \left( \prod_{j=1}^{d} \frac{\text{Pr}(X = x_j | Y = y')}{\text{Pr}(X=\text{ANY} \land Y=y')} + m \right) \frac{\text{Pr}(Y = y')}{\text{Pr}(Y=\text{ANY}) + m}

  = \left( \prod_{j=1}^{d} \frac{C(X=x_j \land Y=y') + mq_j}{C(X=\text{ANY} \land Y=y') + m} \right) \frac{C(Y=y') + mq_y}{C(Y=\text{ANY}) + m}

  – Return the best y’
The Naïve Bayes classifier – v2

- You have a \textit{train} dataset and a \textit{test} dataset
- Initialize an “event counter” (hashtable) \( C \)
- For each example \( id, y, x_1, \ldots, x_d \) in \textit{train}:
  - \( C(“Y=\text{ANY}”)++; \ C(“Y=y”)++ \)
  - For \( j \) in \( 1..d_{id} \):
    • \( C(“Y=y \land X=\text{ANY})++; \ C(“Y=y \land X_j=x_j”)++ \)
- For each example \( id, y, x_1, \ldots, x_d \) in \textit{test}:
  - For each \( y’ \) in \( \text{dom}(Y) \):
    • Compute \( \Pr(y’,x_1,\ldots,x_d) = \)
      \[
      \left( \sum_{j=1}^{d} \log \frac{C(X = x_j \land Y = y’)}{C(X = \text{ANY} \land Y = y’) + m} + mq_j \right) + \log \frac{C(Y = y’)}{C(Y = \text{ANY}) + m} \]
  - Return the best \( y’ \)
The Naïve Bayes classifier – v2

• You have a train dataset and a test dataset

• To classify documents, these might be:
  – http://wcohen.com academic,FacultyHome William W. Cohen Research Professor Machine Learning Department Carnegie Mellon University Member of the Language Technology Institute the joint CMU-Pitt Program in Computational Biology the Lane Center for Computational Biology and the Center for Bioimage Informatics Director of the Undergraduate Minor in Machine Learning Bio Teaching Projects Publications recent all Software Datasets Talks Students Colleagues Blog Contact Info Other Stuff …
  – http://google.com commercial Search Images Videos ….
  – …

• How about for n-grams?
The Naïve Bayes classifier – v2

• You have a *train* dataset and a *test* dataset
• To do C-S spelling correction these might be
  – ng1223 effect a_the b_main d_of e_the
  – ng1224 affect a_shows b_not d_mice e_in
  – ....
• I.e., encode event $X_i = w$ with another event $X = i_w$
• Question: are there any differences in behavior from using A,B,C,D ?
The Naïve Bayes classifier – v2

- You have a train dataset and a test dataset
- Initialize an “event counter” (hashtable) $C$
- For each example $id, y, x_1, \ldots, x_d$ in train:
  - $C(\text{"Y=ANY"})$ ++; $C(\text{"Y=y"})$ ++
  - For $j$ in $1..d_{id}$:
    - $C(\text{"Y=y \land X=ANY"})$ ++; $C(\text{"Y=y \land X_j=x_j"})$ ++
- For each example $id, y, x_1, \ldots, x_d$ in test:
  - For each $y'$ in $\text{dom}(Y)$:
    - Compute $\text{Pr}(y',x_1,\ldots,x_d) = \left( \sum_{j=1}^{d} \log \frac{C(X = x_j \land Y = y') + mq_j}{C(X = \text{ANY} \land Y = y') + m} \right) + \log \frac{C(Y = y') + m q_y}{C(Y = \text{ANY}) + m}$
  - Return the best $y'$

Assume hashtable holding all counts fits in memory

Sequential reads

Complexity: $O(n)$, $n=\text{size of train}$

Sequential reads

Complexity: $O(|\text{dom}(Y)| \times n')$, $n'=\text{size of test}$
Complexity of Naïve Bayes

- You have a *train* dataset and a *test* dataset
- Process:
  - Count events in the *train* dataset
    - \( O(n) \), where \( n \) is total size of *train*
  - Write the counts to disk
    - \( O(\min(|\text{dom}(X)| \times |\text{dom}(Y)|, n)) \)
    - or \( O(|V|) \)
      - if \( V=\text{dom}(X) \) is “vocabulary” and \( \text{dom}(Y) \) is small
  - Read counts, and classify the *test* dataset
    - \( O(|V| + n') \)
  - Worst-case memory usage:
    - \( O(\min(|\text{dom}(X)| \times |\text{dom}(Y)|, n)) \) or \( O(|V|) \)
Naïve Bayes v2

• This is one example of a *streaming classifier*
  – Each example is only read only once
  – You can create a classifier and perform classifications at any point
  – Memory is minimal (<< O(n))
    • Ideally it would be constant
    • Traditionally less than O(sqrt(N))
  – Order doesn’t matter
    • Nice because we may not control the order of examples in real life
      • This is a hard one to get a learning system to have!
• There are few competitive learning methods that as stream-y as naïve Bayes…