CS 188: Artificial Intelligence
Naïve Bayes

Agent Testing Today!

Instructors: Brijen Thananjeyan and Aditya Baradwaj --- University of California, Berkeley

[These slides were created by Dan Klein, Pieter Abbeel, Sergey Levine, with some materials from A. Farhadi. All CS188 materials are at http://ai.berkeley.edu]
Machine Learning

- Up until now: how use a model to make optimal decisions
  - but how do we get the model?
- Machine learning: how to acquire a model from data / experience
  - Learning parameters (e.g. probabilities)
  - Learning structure (e.g. BN graphs)
  - Learning hidden concepts (e.g. clustering)
- Today: model-based classification with Naive Bayes
  - build a model of data (Bayes Net) Then do inference
Classification

Input → Classifier → Output (which category class did input belong to)

Learn this function
Example: Spam Filter

- **Input:** an email
- **Output:** spam/ham

**Setup:**
- Get a large collection of example emails, each labeled “spam” or “ham”
- Note: someone has to hand label all this data!
- Want to learn to predict labels of new, future emails

**Features:** The attributes used to make the ham/spam decision
- Words: FREE!
- Text Patterns: $dd, CAPS
- Non-text: SenderInContacts
- ...

---

Dear Sir.

First, I must solicit your confidence in this transaction, this is by virtue of its nature as being utterly confidencial and top secret. ...

TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

99 MILLION EMAIL ADDRESSES FOR ONLY $99

Ok, I know this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.
Example: Digit Recognition

- **Input:** images / pixel grids
- **Output:** a digit 0-9

**Setup:**
- Get a large collection of example images, each labeled with a digit
- Note: someone has to hand label all this data!
- Want to learn to predict labels of new, future digit images

**Features:** The attributes used to make the digit decision
- Pixels: $(6,8)={\text{ON}}$
- Shape Patterns: NumComponents, AspectRatio, NumLoops
- ...
Other Classification Tasks

- Classification: given inputs $x$, predict labels (classes) $y$

- Examples:
  - Spam detection (input: document, classes: spam / ham)
  - OCR (input: images, classes: characters)
  - Medical diagnosis (input: symptoms, classes: diseases)
  - Automatic essay grading (input: document, classes: grades)
  - Fraud detection (input: account activity, classes: fraud / no fraud)
  - Customer service email routing
  - ... many more

- Classification is an important commercial technology!
1. build model of data from data
2. use model to do classification
Model-Based Classification

- Model-based approach
  - Build a model (e.g. Bayes’ net) where both the label and features are random variables
  - Instantiate any observed features
  - Query for the distribution of the label conditioned on the features

- Challenges
  - What structure should the BN have?
  - How should we learn its parameters?
Naïve Bayes for Digits

- Naïve Bayes: Assume all features are independent effects of the label
  
  weird assumption

- Simple digit recognition version:
  - One feature (variable) $F_{ij}$ for each grid position $<i,j>$
  - Feature values are on / off, based on whether intensity is more or less than 0.5 in underlying image
  - Each input maps to a feature vector, e.g.
    
    \[
    \begin{pmatrix}
    1
    \end{pmatrix}
    \rightarrow \langle F_{0,0} = 0 \ F_{0,1} = 0 \ F_{0,2} = 1 \ F_{0,3} = 1 \ F_{0,4} = 0 \ \ldots \ F_{15,15} = 0 \rangle
    \]
    
  - Here: lots of features, each is binary valued

- Naïve Bayes model: $P(Y|F_{0,0} \ldots F_{15,15}) \propto P(Y) \prod_{i,j} P(F_{i,j}|Y)$

- What do we need to learn? $P(Y)$ and $P(F_{i,j}|Y)$
A general Naive Bayes model:

\[ P(Y, F_1, \ldots, F_n) = P(Y) \prod_{i=1}^{n} P(F_i | Y) \]

This requires a lot of space and data.

The Naive Bayes model is very simplistic, but often works anyway.

We only have to specify how each feature depends on the class. Total number of parameters is \( \frac{n \times |F| \times |Y|}{|Y| \times |F|} \) parameters.

|Y| x |F| x |Y| parameters

\( |F| \) values

\( n \times |F| x |Y| \) parameters

\( |Y| \) parameters

We only have to specify how each feature depends on the class. Total number of parameters is linear in \( n \) instead of \( n \times |Y| \) for the assumption of independence for weird reasons.
Inference for Naïve Bayes

- **Goal:** compute posterior distribution over label variable $Y$
  - **Step 1:** get joint probability of label and evidence for each label

\[
P(Y, f_1 \ldots f_n) = \begin{bmatrix}
P(y_1, f_1 \ldots f_n) \\
P(y_2, f_1 \ldots f_n) \\
\vdots \\
P(y_k, f_1 \ldots f_n)
\end{bmatrix}
\]

- **Step 2:** sum to get probability of evidence

- **Step 3:** normalize by dividing Step 1 by Step 2

\[
P(Y|f_1 \ldots f_n) = \frac{P(Y) \prod_{i} P(f_i|y_1) \cdots P(Y_k) \prod_{i} P(f_i|y_k)}{P(f_1 \ldots f_n)}
\]
General Naïve Bayes

What do we need in order to use Naïve Bayes?

- Inference method (we just saw this part)
  - Start with a bunch of probabilities: \( P(Y) \) and the \( P(F_i | Y) \) tables
  - Use standard inference to compute \( P(Y | F_1...F_n) \)
  - Nothing new here

- Estimates of local conditional probability tables
  - \( P(Y) \), the prior over labels
  - \( P(F_i | Y) \) for each feature (evidence variable)
  - These probabilities are collectively called the **parameters** of the model and denoted by \( q \)
  - Up until now, we assumed these appeared by magic, but...
  - ...they typically come from training data counts: we’ll look at this soon
Example: Conditional Probabilities

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Need to learn these CPTs from data.
A Spam Filter

- Naïve Bayes spam filter
- Data:
  - Collection of emails, labeled spam or ham
  - Note: someone has to hand label all this data!
  - Split into training, held-out, test sets
- Classifiers
  - Learn on the training set
  - (Tune it on a held-out set)
  - Test it on new emails

Dear Sir.
First, I must solicit your confidence in this transaction, this is by virtue of its nature as being utterly confidencial and top secret. …

TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

99 MILLION EMAIL ADDRESSES FOR ONLY $99

Ok, I know this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.
Naïve Bayes for Text

- **Bag-of-words Naïve Bayes:**
  - Features: $W_i$ is the word at position $i$
  - As before: predict label conditioned on feature variables (spam vs. ham)
  - As before: assume features are conditionally independent given label
  - New: each $W_i$ is identically distributed

- **Generative model:**
  \[
P(Y, W_1 \ldots W_n) = P(Y) \prod_i P(W_i|Y)
\]

- **“Tied” distributions and bag-of-words**
  - Usually, each variable gets its own conditional probability distribution $P(F|Y)$
  - In a bag-of-words model
    - Each position is identically distributed
    - All positions share the same conditional probability
    - Why make this assumption?
    - Called “bag-of-words” because model is insensitive to word order or reordering

Word at position $i$, not $i^{th}$ word in the dictionary!

Just need to learn 1 table of words

In is lecture lecture next over person remember room sitting the the the to to up wake when you
Example: Spam Filtering

- Model: \( P(Y, W_1 \ldots W_n) = P(Y) \prod_i P(W_i|Y) \)

- What are the parameters?

| \( P(Y) \) | \( P(W|\text{spam}) \) | \( P(W|\text{ham}) \) |
|---|---|---|
| ham : 0.66 | the : 0.0156 | the : 0.0210 |
| spam : 0.33 | to : 0.0153 | to : 0.0133 |
| | and : 0.0115 | of : 0.0119 |
| | of : 0.0095 | 2002: 0.0110 |
| | you : 0.0093 | with: 0.0110 |
| | a : 0.0086 | from: 0.0108 |
| | with : 0.0080 | and : 0.0107 |
| | from: 0.0075 | a : 0.0105 |
| | ... | ... |

- Where do these tables come from?
Spam Example

| Word | $P(w|\text{spam})$ | $P(w|\text{ham})$ | Tot Spam | Tot Ham |
|------|-------------------|------------------|----------|---------|
| (prior) | 0.33333 | 0.66666 | -1.1 | -0.4 |

$P(\text{spam} | w) = 98.9$
Training and Testing

- **Training set**
  - fit params
  - how well were params fit
  - used to tune learning alg

- **Held-out set**
  - Practice Exam

- **Test set**
  - Final Exam!
  - only once simulates brand new unseen data
Important Concepts

- **Data**: labeled instances, e.g. emails marked spam/ham
  - Training set
  - Held out set
  - Test set
- **Features**: attribute-value pairs which characterize each x
- **Experimentation cycle**
  - Learn parameters (e.g. model probabilities) on training set
  - (Tune hyperparameters on held-out set)
  - Compute accuracy of test set
  - Very important: never “peek” at the test set!
- **Evaluation**
  - Accuracy: fraction of instances predicted correctly
- **Overfitting and generalization**
  - Want a classifier which does well on test data
  - Overfitting: fitting the training data very closely, but not generalizing well
  - Underfitting: fits the training set poorly
Underfitting and Overfitting
Overfitting

Degree 15 polynomial

\[ \text{underfitting, not good either} \]

\[ \text{true function} \]

\[ \text{overfitting: memorizing data, not actually learning} \]
Example: Overfitting

\[ P(\text{features}, C = 2) \]
\[ P(C = 2) = 0.1 \]
\[ P(\text{on}|C = 2) = 0.8 \]
\[ P(\text{on}|C = 2) = 0.1 \]
\[ P(\text{off}|C = 2) = 0.1 \]
\[ P(\text{on}|C = 2) = 0.01 \]

\[ P(\text{features}, C = 3) \]
\[ P(C = 3) = 0.1 \]
\[ P(\text{on}|C = 3) = 0.8 \]
\[ P(\text{on}|C = 3) = 0.9 \]
\[ P(\text{off}|C = 3) = 0.7 \]
\[ P(\text{on}|C = 3) = 0.0 \]
Example: Overfitting

- Postiors determined by *relative* probabilities (odds ratios):

\[
\frac{P(W|\text{ham})}{P(W|\text{spam})} \quad \text{vs} \quad \frac{P(W|\text{spam})}{P(W|\text{ham})}
\]

<table>
<thead>
<tr>
<th>South-west : inf</th>
<th>Screens : inf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nation : inf</td>
<td>Minute : inf</td>
</tr>
<tr>
<td>Morally : inf</td>
<td>Guaranteed : inf</td>
</tr>
<tr>
<td>Nicely : inf</td>
<td>Delivery : inf</td>
</tr>
<tr>
<td>Extent : inf</td>
<td>Signature : inf</td>
</tr>
<tr>
<td>Seriously : inf</td>
<td>...</td>
</tr>
</tbody>
</table>

*What went wrong here?*

*Didn't see all possible words in each class.*
Generalization and Overfitting

- Relative frequency parameters will **overfit** the training data!
  - Just because we never saw a 3 with pixel (15,15) on during training doesn’t mean we won’t see it at test time
  - Unlikely that every occurrence of “minute” is 100% spam
  - Unlikely that every occurrence of “seriously” is 100% ham
  - What about all the words that don’t occur in the training set at all?
  - In general, we can’t go around giving unseen events zero probability

- As an extreme case, imagine using the entire email as the only feature
  - Would get the training data perfect (if deterministic labeling)
  - Wouldn’t *generalize* at all
  - Just making the bag-of-words assumption gives us some generalization, but isn’t enough

- To generalize better: we need to **smooth** or **regularize** the estimates
  - * Bias the parameter fitting process*
Parameter Estimation
Parameter Estimation

- Estimating the distribution of a random variable
- *Elicitation*: ask a human (why is this hard?)
- *Empirically*: use training data (learning!)
  - E.g.: for each outcome $x$, look at the *empirical rate* of that value:
    \[
P_{\text{ML}}(x) = \frac{\text{count}(x)}{\text{total samples}}
    \]
    \[
P_{\text{ML}}(r) = \frac{2}{3}
    \]
  - This is the estimate that maximizes the *likelihood of the data*

\[
L(x, \theta) = \prod_i P_\theta(x_i) = \theta \cdot \theta \cdot (1 - \theta)
\]

\[
P_\theta(x = \text{red}) = \theta
\]
\[
P_\theta(x = \text{blue}) = 1 - \theta
\]
Your First Consulting Job

- A billionaire tech entrepreneur asks you a question:
  - He says: I have thumbtack, if I flip it, what’s the probability it will fall with the nail up?
  - You say: Please flip it a few times:

- You say: The probability is:
  - \( P(H) = \frac{3}{5} \)

- He says: Why???
- You say: Because...
Your First Consulting Job

- $P(\text{Heads}) = \theta$, $P(\text{Tails}) = 1 - \theta$

- Flips are \textit{i.i.d.}: $D = \{x_i | i = 1 \ldots n\}$, $P(D | \theta) = \Pi_i P(x_i | \theta)$
  - Independent events
  - Identically distributed according to unknown distribution

- Sequence $D$ of $\bigvee_H$ Heads and $\bigvee_T$ Tails

$$P(D | \theta) = \theta^{\alpha_H} (1 - \theta)^{\alpha_T}$$
Maximum Likelihood Estimation

- **Data:** Observed set $D$ of $\checkmark_H$ Heads and $\checkmark_T$ Tails
- **Hypothesis space:** Bernoulli distributions
- **Learning:** finding $q$ is an optimization problem
  - What’s the objective function?
    \[
    P(D \mid \theta) = \theta^{\alpha_H} (1 - \theta)^{\alpha_T}
    \]
- **MLE:** Choose $\hat{\theta}$ to maximize probability of $D$
  \[
  \hat{\theta} = \arg \max_\theta P(D \mid \theta) = \arg \max_\theta \ln P(D \mid \theta)
  \]
Maximum Likelihood Estimation

\[ \hat{\theta} = \arg \max_{\theta} \ln P(D \mid \theta) \]

\[ = \arg \max_{\theta} \ln \theta^{\alpha_H} (1 - \theta)^{\alpha_T} \]

- Set derivative to zero, and solve!

\[ \frac{d}{d\theta} \ln P(D \mid \theta) = \frac{d}{d\theta} \left[ \ln \theta^{\alpha_H} (1 - \theta)^{\alpha_T} \right] \]

\[ = \frac{d}{d\theta} \left[ \alpha_H \ln \theta + \alpha_T \ln(1 - \theta) \right] \]

\[ = \alpha_H \frac{d}{d\theta} \ln \theta + \alpha_T \frac{d}{d\theta} \ln(1 - \theta) \]

\[ = \frac{\alpha_H}{\theta} - \frac{\alpha_T}{1 - \theta} = 0 \]

\[ \hat{\theta}_{MLE} = \frac{\alpha_H}{\alpha_H + \alpha_T} \]
Maximum Likelihood?

- Relative frequencies are the maximum likelihood estimates

\[
\theta_{ML} = \arg \max_{\theta} P(X|\theta) \\
= \arg \max_{\theta} \prod_{i} P_{\theta}(X_i) \\
\Rightarrow P_{ML}(x) = \frac{\text{count}(x)}{\text{total samples}}
\]

- Another option is to consider the most likely parameter value given the data

\[
\theta_{MAP} = \arg \max_{\theta} P(\theta|X) \\
= \arg \max_{\theta} \frac{P(X|\theta)P(\theta)}{P(X)} \\
= \arg \max_{\theta} P(X|\theta)P(\theta)
\]

If \( P(\theta) = \text{constant (no bias)} \), we get \( \text{MLE} \)
Unseen Events
Laplace Smoothing

- Laplace’s estimate:
  - Pretend you saw every outcome once more than you actually did

\[
P_{LAP}(x) = \frac{c(x) + 1}{\sum x [c(x) + 1]}
\]

\[
= \frac{c(x) + 1}{N + |X|}
\]

- Can derive this estimate with Dirichlet priors (see cs281a)

\[
P_{ML}(X) = \]

\[
P_{LAP}(X) = \]
Laplace Smoothing

- Laplace’s estimate (extended):
  - Pretend you saw every outcome $k$ extra times

  $$P_{LAP,k}(x) = \frac{c(x) + k}{N + k|X|}$$

  - What’s Laplace with $k = 0$?
  - $k$ is the strength of the prior

  - Laplace for conditionals:
  - Smooth each condition independently:

  $$P_{LAP,k}(x|y) = \frac{c(x, y) + k}{c(y) + k|X|}$$

$r$ $r$ $b$

$P_{LAP,0}(X) =$

$P_{LAP,1}(X) =$

$P_{LAP,100}(X) =$

requires more data to overcome
Real NB: Smoothing

- For real classification problems, smoothing is critical
- New odds ratios:

\[
\frac{P(W|\text{ham})}{P(W|\text{spam})} \quad \frac{P(W|\text{spam})}{P(W|\text{ham})}
\]

| helvetica : 11.4 | verdana : 28.8 |
| seems : 10.8 | Credit : 28.4 |
| group : 10.2 | ORDER  : 27.2 |
| ago : 8.4 | <FONT> : 26.2 |
| areas : 8.3 | money  : 26.5 |
| ... | ... |

Do these make more sense?
Tuning
Tuning on Held-Out Data

Now we’ve got two kinds of unknowns

- Parameters: the probabilities $P(X|Y)$, $P(Y)$
- Hyperparameters: e.g. the amount / type of smoothing to do, $k$, $\sigma$

What should we learn where?

- Learn parameters from training data
- Tune hyperparameters on different data
  - Why? Overfitting
- For each value of the hyperparameters, train and test on the held-out data
- Choose the best value and do a final test on the test data
Bayes rule lets us do diagnostic queries with causal probabilities

- same BN inference as before

The naïve Bayes assumption takes all features to be independent given the class label

- silly, but can work surprisingly well

We can build classifiers out of a naïve Bayes model using training data

- Smoothing estimates is important in real systems

- ble we don't have infinite training data