Up until now: how to reason in a model and how to make optimal decisions

Machine learning: how to select a model on the basis of data / experience
- Learning parameters (e.g. probabilities)
- Learning structure (e.g. BN graphs)
- Learning hidden concepts (e.g. clustering)

Example: Spam Filter
- Input: 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 confidential and top secret.
  - $99 MILLION EMAIL ADDRESSES FOR ONLY $99

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)=ON
  - Shape Patterns: NumComponents, AspectRatio, NumLoops

Other Classification Tasks
- In classification, we predict labels y (classes) for inputs x
- Examples:
  - Spam detection (input: document, classes: spam / ham)
  - OCR (input: images, classes: characters)
  - Medical diagnosis (input: symptoms, classes: diseases)
  - Automatic essay grader (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!

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:
  - Overfitting: fitting the training data very closely, but not generalizing well
  - Generalization: fitting the training data very closely, but not overfitting
  - We’ll investigate overfitting and generalization formally in a few lectures
### Bayes Nets for Classification

- One method of classification:
  - Use a probabilistic model!
  - Features are observed random variables \( F \)
  - \( Y \) is the query variable
  - Use probabilistic inference to compute most likely \( Y \)

\[
y = \arg\max_y P(y|f_1 \ldots f_n)
\]

- You already know how to do this inference

### Simple Classification

- Simple example: two binary features

\[
P(m|s, f) \quad \text{direct estimate}
\]

\[
P(m|s, f) = \frac{P(o, f|m)P(m)}{P(s, f)} \quad \text{Bayes estimate (no assumptions)}
\]

\[
P(m|s, f) = \frac{P(s|m)P(f|m)P(m)}{P(s, f)} \quad \text{Conditional independence}
\]

\[
= \sum \left[ P(m, s, f) = P(s|m)P(f|m)P(m) \right.
\]

\[
\left. P(\bar{m}, s, f) = P(s|\bar{m})P(f|\bar{m})P(\bar{m}) \right]
\]

### General Naïve Bayes

- A general naïve Bayes model:

\[
P(Y, F_1 \ldots F_n) = \frac{P(Y) \prod P(F_i|Y)}{|Y| \text{ parameters} \times n \times |F| \times |Y| \text{ parameters}}
\]

- We only specify how each feature depends on the class
- Total number of parameters is linear in \( n \)

### Inference for Naïve Bayes

- Goal: compute posterior over causes

  - Step 1: get joint probability of causes and evidence

\[
P(Y, f_1 \ldots f_n) =
\]

\[
\left[ P(y_1, f_1 \ldots f_n) \right]
\]

\[
\left[ P(y_2, f_1 \ldots f_n) \right]
\]

\[
P(y_n, f_1 \ldots f_n)
\]

\[
P(f_1) \prod P(f_i|y_i) \]

\[
P(f_1) \prod P(f_i|y_i)
\]

\[
P(f_1) \prod P(f_i|y_i)
\]

\[
P(Y|f_1 \ldots f_n)
\]

- Step 2: get probability of evidence
- Step 3: renormalize

### General Naïve Bayes

- What do we need in order to use naïve Bayes?

  - Inference (you know this part)
    - Start with a bunch of conditionals, \( P(Y) \) and the \( P(F_i|Y) \) tables
    - Use standard inference to compute \( P(Y|f_1 \ldots f_n) \)
    - 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 \( \theta \)
    - Up until now, we assumed these appeared by magic, but...
    - ...they typically come from training data: we’ll look at this now

### A Digit Recognizer

- Input: pixel grids

- Output: a digit 0-9
Naïve Bayes for Digits

- **Simple version:**
  - One feature $F_{ij}$ for each grid position $<i,j>$
  - Possible 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.
    $$P(Y|F_{0,0} \cdots F_{15,15}) \propto P(Y) \prod_{i,j} P(F_{ij}|Y)$$
- What do we need to learn?

Parameter Estimation

- Estimating distribution of random variables like $X$ or $X | Y$
- **Empirically:** use training data
  - For each outcome $x$, look at the empirical rate of that value:
    $$P_{ML}(x) = \frac{\text{count}(x)}{\text{total samples}}$$
  - This is the estimate that maximizes the likelihood of the data
    $$L(x, \theta) = \prod_i P_\theta(x_i)$$
- **Elicitation:** ask a human!
  - Usually need domain experts, and sophisticated ways of eliciting probabilities (e.g. betting games)
  - Trouble calibrating

Naïve Bayes for Text

- Bag-of-Words Naïve Bayes:
  - Predict unknown class label (spam vs. ham)
  - Assume evidence features (e.g. the words) are independent
  - Warning: subtly different assumptions than before!
- **Generative model**
  $$P(C, W_1 \ldots W_n) = P(Y) \prod_{i} P(W_i|C)$$
- 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 probs $P(W|C)$
  - Why make this assumption?

Examples: CPTs

| $P(Y)$ | $P(F_{0,0} = \text{on}|Y)$ | $P(F_{0,5} = \text{on}|Y)$ |
|--------|-----------------------------|-----------------------------|
| 1      | 0.1                         | 1                           |
| 2      | 0.1                         | 2                           |
| 3      | 0.1                         | 3                           |
| 4      | 0.1                         | 4                           |
| 5      | 0.1                         | 5                           |
| 6      | 0.1                         | 6                           |
| 7      | 0.1                         | 7                           |
| 8      | 0.1                         | 8                           |
| 9      | 0.1                         | 9                           |
| 0      | 0.1                         | 0                           |

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

Example: Spam Filtering

- **Model:**
  $$P(C, W_1 \ldots W_n) = P(C) \prod_i P(W_i|C)$$
- **What are the parameters?**

| $P(C)$ | $P(W|\text{spam})$ | $P(W|\text{ham})$ |
|--------|---------------------|---------------------|
| $\text{ham}$ | 0.66 | $\text{the}$: 0.0156 | $\text{the}$: 0.0210 |
| $\text{spam}$ | 0.33 | $\text{to}$: 0.0153 | $\text{to}$: 0.0138 |
| | | $\text{and}$: 0.0115 | $\text{of}$: 0.0119 |
| | | $\text{you}$: 0.0093 | $\text{2002}$: 0.0110 |
| | | $\text{a}$: 0.0086 | $\text{with}$: 0.0108 |
| | | $\text{from}$: 0.0080 | $\text{from}$: 0.0107 |
| | | $\text{and}$: 0.0105 | $\text{and}$: 0.0105 |
| | | $\text{of}$: 0.0100 | $\text{of}$: 0.0100 |
| | | $\text{...}$ | $\text{...}$ |

- Where do these tables come from?
Spam Example

| Word   | P(w|spam) | P(w|ham)       | Tot Spam | Tot Ham |
|--------|----------|---------------|----------|---------|
| prior  | 0.33333  | 0.66666       | -1.1     | -0.4    |

\[ P(\text{spam} \mid w) = 98.9 \]

Example: Overfitting

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

2 wins!!

Example: Spam Filtering

- Raw probabilities alone don’t affect the posteriors; relative probabilities (odds ratios) do:
  \[ \frac{P(W \mid \text{spam})}{P(W \mid \text{ham})} \]

| south-west : inf | screens : inf |
| nation : inf | minute : inf |
| morally : inf | guaranteed : inf |
| nicely : inf | $205.00 : inf |
| extent : inf | delivery : inf |
| seriously : inf | signature : inf |
| ... | ...

What went wrong here?

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

Estimation: Smoothing

- Problems with maximum likelihood estimates:
  - If I flip a coin once, and it’s heads, what’s the estimate for \( P(\text{heads})? \)
  - What if I flip 10 times with 8 heads?
  - What if I flip 10M times with 8M heads?
- Basic idea:
  - We have some prior expectation about parameters (here, the probability of heads)
  - Given little evidence, we should skew towards our prior
  - Given a lot of evidence, we should listen to the data

### Estimation: Smoothing

- Relative frequencies are the maximum likelihood estimates
  \[ \theta_{\text{MLE}} = \arg \max_{\theta} P(X \mid \theta) \]
  \[ P_{\text{MLE}}(x) = \frac{\text{count}(x)}{\text{total samples}} \]

- In Bayesian statistics, we think of the parameters as just another random variable, with its own distribution
  \[ \theta_{\text{MAP}} = \arg \max_{\theta} P(\theta \mid X) \]
  \[ = \arg \max_{\theta} P(\theta ; P(\theta)) = P(X \mid \theta) \]

\[ ??? \]
Estimation: Laplace Smoothing

- **Laplace’s estimate**:
  - Pretend you saw every outcome once more than you actually did
  
  \[
  P_{\text{LAP}}(x) = \frac{c(x) + 1}{\sum_x (c(x) + 1)}
  \]
  
  Can derive this as a MAP estimate with Dirichlet priors (see cs281a)

- **Laplace’s estimate (extended)**:
  - Pretend you saw every outcome \(k\) extra times
  
  \[
  P_{\text{LAP}}(x) = \frac{c(x) + k}{\sum_x (c(x) + k)}
  \]
  
  What’s Laplace with \(k = 0\)?
  
  \[
  P_{\text{LAP}}(x) = \frac{c(x)}{\sum_x c(x)}
  \]
  
  \(k\) is the strength of the prior

- **Laplace for conditionals**: Smooth each condition independently:
  
  \[
  P_{\text{LAP}_k}(x|y) = \frac{c(x, y) + k}{c(y) + k|x|}
  \]

Estimation: Linear Interpolation

- In practice, Laplace often performs poorly for \(P(X|Y)\):
  - When \(|X|\) is very large
  - When \(|Y|\) is very large

- Another option: linear interpolation
  - Also get \(P(X)\) from the data
  - Make sure the estimate of \(P(X|Y)\) isn’t too different from \(P(X)\)
  
  \[
  P_{\text{LIN}}(x|y) = \alpha \hat{P}(x|y) + (1.0 - \alpha) \hat{P}(x)
  \]

- What if \(\alpha\) is 0? 1?

- For even better ways to estimate parameters, as well as details of the math see cs281a, cs288

Real NB: Smoothing

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

  \[
  \begin{aligned}
  P(W|\text{ham}) & \quad P(W|\text{spam}) \\
  P(W|\text{spam}) & \quad P(W|\text{ham})
  \end{aligned}
  \]

  \[
  \begin{array}{l}
  \text{helvetica} : 11.4 \\
  \text{seems} : 10.8 \\
  \text{group} : 10.2 \\
  \text{areas} : 8.3 \\
  \text{...} \\
  \text{verdana} : 28.8 \\
  \text{Credit} : 28.4 \\
  \text{ORDER} : 27.2 \\
  \text{money} : 26.5 \\
  \text{...}
  \end{array}
  \]

  Do these make more sense?

Tuning on Held-Out Data

- Now we’ve got two kinds of unknowns
  - Parameters: the probabilities \(P(Y|X), P(Y)\)
  - Hyperparameters, like the amount of smoothing to do: \(k, \alpha\)

- Where to learn?
  - Learn parameters from training data
  - Must tune hyperparameters on different data

  - Why?
    - 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 held-out data

Baselines

- First task: get a baseline
  - Baselines are very simple “straw man” procedures
  - Help determine how hard the task is
  - Help know what a “good” accuracy is

- Weak baseline: most frequent label classifier
  - Gives all test instances whatever label was most common in the training set
  - E.g. for spam filtering, might label everything as ham
  - Accuracy might be very high if the problem is skewed

- For real research, usually use previous work as a (strong) baseline
Confidences from a Classifier

- The confidence of a probabilistic classifier:
  \[ \text{confidence}(\mathbf{x}) = \arg \max \ P(y|x) \]
- Represents how sure the classifier is of the classification
- Any probabilistic model will have confidences
- No guarantee confidence is correct

Calibration

- Weak calibration: higher confidences mean higher accuracy
- Strong calibration: confidence predicts accuracy rate
- What's the value of calibration?

Errors, and What to Do

- Examples of errors

What to Do About Errors?

- Need more features—words aren’t enough!
- Have you emailed the sender before?
- Have 1K other people just gotten the same email?
- Is the sending information consistent?
- Is the email in ALL CAPS?
- Do inline URLs point where they say they point?
- Does the email address you by (your) name?

- Can add these information sources as new variables in the NB model
- Next class we’ll talk about classifiers which let you easily add arbitrary features more easily

Summary

- Bayes rule lets us do diagnostic queries with causal probabilities
- The naive Bayes assumption takes all features to be independent given the class label
- We can build classifiers out of a naive Bayes model using training data
- Smoothing estimates is important in real systems
- Classifier confidences are useful, when you can get them

Case-Based Reasoning

- Similarity for classification
  - Case-based reasoning
  - Predict an instance’s label using similar instances
- Nearest-neighbor classification
  - 1-NN: copy the label of the most similar data point
  - K-NN: let the k nearest neighbors vote (have to devise a weighting scheme)
- Key issue: how to define similarity
- Trade-off:
  - Small k gives relevant neighbors
  - Large k gives smoother functions
  - Sound familiar?
- [DEMO]
Recap: Nearest-Neighbor

- Nearest neighbor:
  - Classify test example based on closest training example
  - Requires a similarity function (kernel)
  - Eager learning: extract classifier from data
  - Lazy learning: keep data around and predict from it at test time

2 Examples 10 Examples 100 Examples 10000 Examples

Truth

Nearest-Neighbor Classification

- Nearest neighbor for digits:
  - Take new image
  - Compare to all training images
  - Assign based on closest example

- Encoding: image is vector of intensities:
  \[ \mathbf{1} = (0.0 \ 0.0 \ 0.3 \ 0.6 \ 0.7 \ 0.1 \ldots 0.0) \]

- What’s the similarity function?
  - Dot product of two images vectors?
  \[ \text{sim}(x, y) = x \cdot y = \sum_i x_i y_i \]
  - Usually normalize vectors so \( ||x|| = 1 \)
  - \( \min = 0 \) (when?), \( \max = 1 \) (when?)