CMPT-825
Natural Language Processing

Anoop Sarkar
http://www.cs.sfu.ca/∼anoop

November 15, 2010

Probability Models

- $p(x, y)$: x = input, y = labels
- Pick best prob distribution $p(x, y)$ to fit the data
- Max likelihood of the data *according to the prob model* equivalent to minimizing entropy

Probability Models

- Max likelihood of the data *according to the prob model*
- Equivalent to picking best parameter values θ such that the data gets highest likelihood:

$$\max_\theta p(\theta \mid \text{data}) = \max_\theta p(\theta) \times p(\text{data} \mid \theta)$$

Log probabilities v.s. scores

- n-grams: $\ldots + \log\, p(w_8 \mid w_6, w_7) + \ldots$
- HMM: $\ldots + \log\, p(t_5 \mid t_4) + \log\, p(w_5 \mid t_5) + \ldots$
- Naive Bayes: $\log\, p(\text{class}) + \log\, p(\text{feature}_1 \mid \text{class}) + \log\, p(\text{feature}_2 \mid \text{class}) + \ldots$

Advantages of probability models

- parameters can be estimated automatically, while scores have to twiddled by hand
- parameters can be estimated from supervised or unsupervised data
- probabilities can be used to quantify confidence in a particular state and used to compare against other probabilities in a strictly comparable setting
- modularity: $p(semantics) \times p(syntax \mid semantics) \times p(morphology \mid syntax) \times p(phonology \mid morphology) \times p(sounds \mid phonology)$

Remember the humble Naive Bayes Classifier

- $\mathbf{x}$ is the input that can be represented as d independent features f_j, $1 \leq j \leq d$
- y is the output classification
- $P(y \mid \mathbf{x}) = \frac{P(y) \times P(\mathbf{x}|y)}{P(\mathbf{x})}$
- $P(\mathbf{x} \mid y) = \prod_{j=1}^{d} P(f_j \mid y)$
- $P(y \mid \mathbf{x}) = P(y) \times \prod_{j=1}^{d} P(f_j \mid y)$

Using Naive Bayes for Document Classification

- Spam text: `Learn how to make $38.99 into a money making machine that pays ... $7,000 / month !`
- Distinguish spam text from regular email text
- Find useful features to make this distinction

Using Naive Bayes

- Useful features
  1. contains `turn $AMOUNT into`
  2. contains `$AMOUNT`
  3. contains `Learn how to`
  4. contains exclamation mark at end of sentence

Using Naive Bayes

- how many times do these features occur?
  1. contains: `turn $AMOUNT into`
     in spam text: 50
     in normal email: 2
     i.e. 25x more likely in spam
  2. contains: `$AMOUNT`
     in spam text: 90
     in normal email: 10
     i.e. 9x more likely in spam

Using Naive Bayes

- How likely is it for *both* features to occur at the same time in a spam message?
  1. contains: `turn $AMOUNT into`
  2. contains: `$AMOUNT`
- Assume we have a new feature, contains: `turn $AMOUNT into` *and* `$AMOUNT`
- The model predicts that the event that both features occur simultaneously has probability $\frac{140}{152} = 0.92$
- But Naive Bayes assumes that these features are independent and should occur with probability: $0.92 \cdot 0.9 = 0.864$

Using Naive Bayes

- Naive Bayes needs overlapping but independent features
- How can we use all of the features we want?
  1. contains `turn $AMOUNT into`
  2. contains `$AMOUNT`
  3. contains `Learn how to`
  4. contains exclamation mark at end of sentence
- how about giving each feature a weight w equal to its log probability: $w = \log p(f, y)$

Using Naive Bayes

- each feature gets a score equal to its log probability
- Assign scores to features:
  1. $w_1 = +1$ contains `turn $AMOUNT into`
  2. $w_2 = +5$ contains `$AMOUNT`
  3. $w_3 = +0.2$ contains `Learn how to`
  4. $w_4 = -2$ contains exclamation mark at end of sentence

Using Naive Bayes

- so add the scores and treat it like a log probability
- $\log p(\mathit{spam} \mid \mathit{feats}) = 4.2$
- but then, $p(\mathit{spam} \mid \mathit{feats}) = \mathit{exp}(4.2) = 66.68$
- how do we compute keep arbitrary scores and still get probabilities?

Log linear model

- Let there be m features, $f_k(\mathbf{x}, y)$ for $k = 1, \ldots, m$
- Define a parameter vector $\mathbf{w} \in \mathbb{R}^m$
- Each $(\mathbf{x}, y)$ pair is mapped to score:

$$s(\mathbf{x}, y) = \sum_k w_k \cdot f_k(\mathbf{x}, y)$$

- Using inner product notation:

$$\begin{aligned} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, y) &= \sum_k w_k \cdot f_k(\mathbf{x}, y) \\ s(\mathbf{x}, y) &= \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, y) \end{aligned}$$

- To get a probability from the score: Renormalize!

$$\Pr(y \mid \mathbf{x}, \mathbf{w}) = \frac{exp\left(s(\mathbf{x}, y)\right)}{\sum_{y'} exp\left(s(\mathbf{x}, y')\right)}$$

Log linear model

- The name 'log-linear model' comes from:

$$\log \Pr(y \mid \mathbf{x}, \mathbf{w}) = \underbrace{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, y)}_{\text{linear term}} - \underbrace{\log \sum_{y'} exp\left(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, y')\right)}_{\text{normalization term}}$$

- Once the weights are learned, we can perform predictions using these features.
- The goal: to find $\mathbf{w}$ that maximizes the log likelihood $L(\mathbf{w})$ of the labeled training set containing $(\mathbf{x}_i, y_i)$ for $i = 1 \dots n$

$$\begin{aligned} L(\mathbf{w}) &= \sum_i \log \Pr(y_i \mid \mathbf{x}_i, \mathbf{w}) \\ &= \sum_i \mathbf{w} \cdot \mathbf{f}(\mathbf{x}_i, y_i) - \sum_i \log \sum_{y'} exp\left(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}_i, y')\right) \end{aligned}$$

Log linear model

- Maximize:

$$L(\mathbf{w}) = \sum_i \mathbf{w} \cdot \mathbf{f}(\mathbf{x}_i, y_i) - \sum_i \log \sum_{y'} exp\left(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}_i, y')\right)$$

- Calculate gradient:

$$\left.\frac{dL(\mathbf{w})}{d\mathbf{w}}\right|_{\mathbf{w}}$$

$$= \sum_i \mathbf{f}(\mathbf{x}_i, y_i) - \sum_i \frac{1}{\sum_{y''} exp\left(\mathbf{w} \cdot \mathbf{f}(x_i, y'')\right)} \sum_{y'} \mathbf{f}(\mathbf{x}_i, y') \cdot exp\left(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}_i, y')\right)$$

$$= \sum_i \mathbf{f}(\mathbf{x}_i, y_i) - \sum_i \sum_{y'} \mathbf{f}(\mathbf{x}_i, y') \frac{exp\left(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}_i, y')\right)}{\sum_{y''} exp\left(\mathbf{w} \cdot \mathbf{f}(x_i, y'')\right)}$$

$$= \underbrace{\sum_i \mathbf{f}(\mathbf{x}_i, y_i)}_{\text{Observed counts}} - \underbrace{\sum_i \sum_{y'} \mathbf{f}(\mathbf{x}_i, y') \Pr(y' \mid \mathbf{x}_i, \mathbf{w})}_{\text{Expected counts}}$$

Log linear model

- Init: $\mathbf{w}^{(0)} = \mathbf{0}$
- $t \leftarrow 0$
- Iterate until convergence:
  - Calculate: $\Delta = \left.\frac{dL(\mathbf{w})}{d\mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}^{(t)}}$
  - Find $\beta^* = \text{argmax}_\beta \, L(\mathbf{w}^{(t)} + \beta\Delta)$
  - Set $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \beta^*\Delta$

Learning the weights: **w**: Generalized Iterative Scaling

$f^{\#} = max_{x,y} \sum_{j=1}^{k} f_j(x, y)$

For each iteration

expected[1 .. # of features] $\leftarrow 0$

For i = 1 to $|$ training data $|$

For each feature f_j

expected[j] $+= f_j(x_i, y_i) \times P(y_i \mid x_i)$

For each feature f_j

observed[j] $= f_j(x, y) \times \frac{c(x,y)}{|\text{training data}|}$

For each feature f_j

$w_j \leftarrow w_j \times \sqrt[f^{\#}]{\frac{\text{observed[j]}}{\text{expected[j]}}}$

cf. Goodman, NIPS '01

Maximum Entropy

- The maximum entropy principle: related to Occam's razor and other similar justifications for scientific inquiry
- Make the minimum possible assumptions about unseen data
- Also: Laplace's *Principle of Insufficient Reason*: when one has no information to distinguish between the probability of two events, the best strategy is to consider them equally likely

Logistic Regression

- models effects of explanatory variables on binary valued variable
- observations $\mathbf{x} = \{x_1, \ldots, x_j\}$ with success given by $q(\mathbf{x})$:

$$q(\mathbf{x}) = \frac{e^{g(\mathbf{x})}}{1 + e^{g(\mathbf{x})}}$$

and

$$g(\mathbf{x}) = \beta_0 + \sum_{j=1}^{k} \beta_j x_j$$

Logistic Regression

- probability that observations lead to success, or $p(a = 1 \mid b)$:

$$p(a = 1 \mid b) = \frac{e^{g(b)}}{1 + e^{g(b)}}$$

where

$$g(b) = \beta_0 f_0(1, b) + \sum_{j=1}^{k} \beta_j f_j(1, b)$$

- $\beta_j = \log \alpha_j, \; f_0(1, b) = 1$ and $f_j(1, b) = x_j$