Outline

1. Administration
2. Review of last lecture
3. Logistic regression
A few announcements

- Homework 1: ==========
- Homework 1: due 9/24 (see the homework sheets for detailed submission information)
- A lot of goodies on the Discussion Board: please participate or at least browse!
- Typos (corrected versions uploaded already or soon)
  - QA-problem: Hardcopy due 9/24 at 5pm CS Front Desk
  - Programming-problem: Electronic version due 9/24 11:59pm on Blackboard
- A lot of goodies on the Discussion Board: please participate or at least browse!
Outline

1. Administration

2. Review of last lecture
   - Naive Bayes

3. Logistic regression
Naive Bayes

Assume $X \in \mathbb{R}^D$ and all $X_d \in [K]$

$$P(X = x, Y = c) = P(Y = c) \prod_k P(k|Y = c)^{z_k} = \pi_c \prod_k \theta^{z_k}_{ck}$$

where $z_k$ is the number of times $k$ in $x$.

Key assumption made

- Conditional independence:
  $$P(X_i, X_j|Y = c) = P(X_i|Y = c)P(X_j|Y = c).$$

- $P(X_i|Y = c)$ depends only the value of $X_i$, not $i$ itself (order of words does not matter in “bag-of-word” representation of documents)
Special case (i.e., our model of spam emails)

Assumptions

- All $X_d$ are categorical variables from the same domain — $x_d \in [K]$, for example, the index to the unique words in a dictionary.
- $P(X_d = x_d | Y = y)$ depends only on the value of $x_d$, not $d$ itself, namely, orders are not important (thus, we only need to count).

Simplified definition

$$P(X = x, Y = c) = P(Y = c) \prod_k P(k | Y = c)^{z_k} = \pi_c \prod_k \theta_{ck}^{z_k}$$

where $z_k$ is the number of times $k$ in $x$.

Note that we only need to enumerate in the product, the index to the $x_d$'s possible values. On the previous slide, however, we enumerate over $d$ as we do not have the assumption there that order is not important.
Learning problem

Training data

\[ \mathcal{D} = \{ (x_n, y_n) \}_{n=1}^N \rightarrow \mathcal{D} = \{ (\{z_{nk}\}_{k=1}^K, y_n) \}_{n=1}^N \]

Goal
Learn \( \pi_c, c = 1, 2, \cdots, C, \) and \( \theta_{ck}, \forall c \in [C], k \in [K] \) under the constraint

\[ \sum_c \pi_c = 1 \]

and

\[ \sum_k \theta_{ck} = \sum_k P(k|Y = c) = 1 \]

as well as those quantities should be nonnegative.
Likelihood Function

Let $X_1, \ldots, X_N$ be IID with PDF $f(x|\theta)$ (also written as $f(x; \theta)$). The likelihood function is defined by $L(\theta|x)$ (also written as $L(\theta; x)$),

$$L(\theta|x) = \prod_{i=1}^{N} f(X_i; \theta).$$

**Notes** The likelihood function is just the joint density of the data, except that we treat it as a function of the parameter $\theta$, $L : \Theta \to [0, \infty)$. It is not a density function in general; it does not necessarily integrate to 1 with respect to $\theta$. 
Maximum Likelihood Estimator

**Definition**: The maximum likelihood estimator (MLE) $\hat{\theta}$, is the value of $\theta$ that maximizes $L(\theta)$.

The log-likelihood function is defined by $l(\theta) = \log L(\theta)$. Its maximum occurs at the same place as that of the likelihood function.

The same is true of the likelihood function times any constant. Thus we shall often drop constants in the likelihood function.

The log-likelihood is also sometimes called the *cross-entropy* or *deviance* in the context of classification.
Our hammer: maximum likelihood estimation

Log-Likelihood of the training data

\[ \mathcal{L} = \log P(\mathcal{D}) = \log \prod_{n=1}^{N} \pi_{yn} P(x_n|y_n) \]  \hspace{1cm} (1)

\[ = \log \prod_{n=1}^{N} \left( \pi_{yn} \prod_{k} \theta_{ynk}^{z_{nk}} \right) \]  \hspace{1cm} (2)

\[ = \sum_{n} \left( \log \pi_{yn} + \sum_{k} z_{nk} \log \theta_{ynk} \right) \]  \hspace{1cm} (3)

\[ = \sum_{n} \log \pi_{yn} + \sum_{n,k} z_{nk} \log \theta_{ynk} \]  \hspace{1cm} (4)
Our hammer: maximum likelihood estimation

Log-Likelihood of the training data

\[ \mathcal{L} = \log P(\mathcal{D}) = \log \prod_{n=1}^{N} \pi_{yn} P(x_n | y_n) \]  

\[ = \log \prod_{n=1}^{N} \left( \pi_{yn} \prod_{k} \theta_{ynk}^{z_{nk}} \right) \]  

\[ = \sum_{n} \left( \log \pi_{yn} + \sum_{k} z_{nk} \log \theta_{ynk} \right) \]  

\[ = \sum_{n} \log \pi_{yn} + \sum_{n,k} z_{nk} \log \theta_{ynk} \]

Optimize it!

\[ (\pi^*_c, \theta^*_{ck}) = \arg \max \sum_{n} \log \pi_{yn} + \sum_{n,k} z_{nk} \log \theta_{ynk} \]
Details

Note the separation of parameters in the likelihood

\[ \sum_n \log \pi_{y_n} + \sum_{n,k} z_{nk} \log \theta_{y_n,k} \]

which implies that \( \{\pi_c\} \) and \( \{\theta_{ck}\} \) can be estimated separately.

Reorganize terms

\[ \sum_n \log \pi_{y_n} = \sum_c \log \pi_c \times (\# \text{ of data points labeled as } c) \]

and

\[ \sum_{n,k} z_{nk} \log \theta_{y_n,k} = \sum_c \sum_{n:y_n=c} \sum_k z_{nk} \log \theta_{ck} = \sum_c \sum_{n:y_n=c,k} z_{nk} \log \theta_{ck} \]

The later implies \( \{\theta_{ck}, k = 1, 2, \cdots, K\} \) and \( \{\theta_{c'k}, k = 1, 2, \cdots, K\} \) can be estimated independently.
Estimating $\{\pi_c\}$

We want to maximize

$$\sum_c \log \pi_c \times (\# \text{of data points labeled as } c)$$

Intuition

- Similar to roll a dice (or flip a coin): each side of the dice shows up with a probability of $\pi_c$ (total C sides)
- And we have total N trials of rolling this dice

Solution

$$\pi_c^* = \frac{\# \text{of data points labeled as } c}{N}$$
Estimating \( \{\theta_{ck}, k = 1, 2, \cdots, K\} \)

We want to maximize

\[
\sum_{n: y_n = c, k} z_{nk} \log \theta_{ck}
\]

Intuition
- Similar to roll a dice with color \( c \): each side of the dice shows up with a probability of \( \theta_{ck} \) (total K slides)
- And we have total \( \sum_{n: y_n = c, k} z_{nk} \) trials.

Solution

\[
\theta^*_{ck} = \frac{\text{#of side-k shows up in data points labeled as } c}{\text{#of all slides in data points labeled as } c}
\]
Translating back to our problem of detecting spam emails

- Collect a lot of ham and spam emails as training examples
- Estimate the “bias”
  \[ p(\text{ham}) = \frac{\# \text{ of ham emails}}{\# \text{ of emails}} , \quad p(\text{spam}) = \frac{\# \text{ of spam emails}}{\# \text{ of emails}} \]
- Estimate the weights (i.e., \( p(\text{dollar} | \text{ham}) \) etc)
  \[ p(\text{funny\_word} | \text{ham}) = \frac{\# \text{ of funny\_word in ham emails}}{\# \text{ of words in ham emails}} \quad (5) \]
  \[ p(\text{funny\_word} | \text{spam}) = \frac{\# \text{ of funny\_word in spam emails}}{\# \text{ of words in spam emails}} \quad (6) \]
A short derivation of the maximum likelihood estimation

The steps are similar to the ones in Math Review
To maximize

$$\sum_{n:y_n=c} z_{nk} \log \theta_{ck}$$

We use the Lagrangian multiplier

$$\sum_{n:y_n=c,k} z_{nk} \log \theta_{ck} + \lambda \left( \sum_k \theta_{ck} - 1 \right)$$

Taking derivatives with respect to $\theta_{ck}$ and then find the stationary point

$$\sum_{n:y_n=c} \frac{z_{nk}}{\theta_{ck}} + \lambda = 0 \rightarrow \theta_{ck} = -\frac{1}{\lambda} \sum_{n:y_n=c,k} z_{nk}$$

Apply the constraint that $\sum_k \theta_{ck} = 1$

$$\theta_{ck} = \frac{\sum_{n:y_n=c,k} z_{nk}}{\sum_k \sum_{n:y_n=c} z_{nk}}$$
Classification rule

Given an unlabeled data point \( x = \{ z_k, k = 1, 2, \cdots, K \} \), label it with

\[
y^* = \arg \max_{c \in [C]} P(y = c | x)
\]

\[
= \arg \max_{c \in [C]} P(y = c) P(x | y = c)
\]

\[
= \arg \max_c \left[ \log \pi_c + \sum_k z_k \log \theta_{ck} \right]
\]
Moving forward

Examine the classification rule for naive Bayes

\[ y^* = \arg \max_c \log \pi_c + \sum_k z_k \log \theta_{ck} \]

For binary classification problem, this is just to determine the label basing on

\[ \log \pi_1 + \sum_k z_k \log \theta_{1k} - \left( \log \pi_2 + \sum_k z_k \log \theta_{2k} \right) \]

This is just a linear function of the features \( \{z_k\} \)

\[ w_0 + \sum_k z_k w_k \]

where we “absorb” \( w_0 = \log \pi_1 - \log \pi_2 \) and \( w_k = \log \theta_{1k} - \log \theta_{2k} \).
Naive Bayes is a linear classifier

Fundamentally, what really matters in deciding decision boundary is

$$w_0 + \sum_k z_k w_k$$

This motivates many new methods. One of them is logistic regression, to be discussed next.
You should know or be able to

- What naive Bayes model is
  - write down the joint distribution
  - explain the conditional independence assumption implied by the model
  - explain how this model can be used to distinguish spam from ham emails

- Be able to go through the short derivation for parameter estimation
  - The model illustrated here is called discrete Naive Bayes
  - Your homework asks you to apply the same principle to Gaussian naive Bayes
  - The derivation is very similar – except there you need to estimate Gaussian continuous random variables (instead of estimating discrete random variables like rolling a dice)

- Think about another classification task that this model might be useful
To enhance your understanding

write a personalized spam email detector yourself

- Collect from your own email inbox, 500 samples of spam and good emails (the more, the merrier)
- Create a training (400 samples), validation (50 samples) and test dataset (50 samples)
- Estimate Naive Bayes model parameters for distinguishing ham and spam emails
- Apply the model to classify test dataset (you will use validation dataset later)
- Report your results on Discussion forum and post your questions of doing this experiment

This recipe is not 100% bullet-proof. You will discover practical issues. Working on those issues will improve your understanding of the algorithm and its practice.
Outline

1. Administration

2. Review of last lecture

3. Logistic regression
   - General setup
   - Maximum likelihood estimation
Logistic classification

Setup for two classes

- Input: $\mathbf{x} \in \mathbb{R}^D$
- Output: $y \in \{0, 1\}$
- Training data: $\mathcal{D} = \{(\mathbf{x}_n, y_n), n = 1, 2, \ldots, N\}$
- Model:

  $$p(y = 1|\mathbf{x}; b, \mathbf{w}) = \sigma[g(\mathbf{x})]$$

  where

  $$g(\mathbf{x}) = b + \sum_d w_d x_d = b + \mathbf{w}^T \mathbf{x}$$

  and $\sigma[\cdot]$ stands for the sigmoid function

  $$\sigma(a) = \frac{1}{1 + e^{-a}}$$
Why the sigmoid function?

What does it look like?

\[
\sigma(a) = \frac{1}{1 + e^{-a}}
\]

where

\[
a = b + w^T x
\]
Why the sigmoid function?

What does it look like?

\[ \sigma(a) = \frac{1}{1 + e^{-a}} \]

where

\[ a = b + w^T x \]

Properties

- Bounded between 0 and 1 \( \leftrightarrow \) thus, interpretable as probability
- Monotonically increasing \( \leftrightarrow \) thus, usable to derive classification rules
  1. \( \sigma(a) > 0.5 \), positive (classify as '1')
  2. \( \sigma(a) < 0.5 \), negative (classify as '0')
  3. \( \sigma(a) = 0.5 \), undecidable
- Nice computationally properties These will unfold in the next few slides
Linear or nonlinear?

\( \sigma(a) \) \textbf{is nonlinear}, however, the decision boundary is determined by

\[
\sigma(a) = 0.5 \implies a = 0 \implies g(x) = b + w^T x = 0
\]

which is a \textit{linear} function in \( x \)

We often call \( b \) the bias term.
Contrast Naive Bayes and our new model

**Similar**

Both look at the linear function of features for classification.

**Difference**

Naive Bayes models the *joint* distribution

\[ P(X, Y) = P(Y)P(X|Y) \]

Logistic regression models the *conditional* distribution

\[ P(Y|X) \]
Likelihood function

Probability of a single training sample \((x_n, y_n)\)

\[
p(y_n | x_n; b; w) = \begin{cases} 
\sigma(b + w^T x_n) & \text{if } y_n = 1 \\
1 - \sigma(b + w^T x_n) & \text{otherwise}
\end{cases}
\]
Likelihood function

Probability of a single training sample \((x_n, y_n)\)

\[
p(y_n|x_n; b; w) = \begin{cases} 
\sigma(b + w^T x_n) & \text{if } y_n = 1 \\
1 - \sigma(b + w^T x_n) & \text{otherwise}
\end{cases}
\]

Compact expression, exploring that \(y_n\) is either 1 or 0

\[
p(y_n|x_n; b; w) = \sigma(b + w^T x_n)^y_n [1 - \sigma(b + w^T x_n)]^{1-y_n}
\]
Cross-entropy error

Log-likelihood of the whole training data $\mathcal{D}$

$$
\log P(\mathcal{D}) = \sum_{n} \left\{ y_n \log \sigma(b + \mathbf{w}^T \mathbf{x}_n) + (1 - y_n) \log [1 - \sigma(b + \mathbf{w}^T \mathbf{x}_n)] \right\}
$$
Cross-entropy error

Log-likelihood of the whole training data $\mathcal{D}$

$$\log P(\mathcal{D}) = \sum_{n} \{ y_n \log \sigma(b + \mathbf{w}^{T} \mathbf{x}_n) + (1 - y_n) \log[1 - \sigma(b + \mathbf{w}^{T} \mathbf{x}_n)] \}$$

It is convenient to work with its negation, which is called

cross-entropy error function

$$\mathcal{E}(b, \mathbf{w}) = -\sum_{n} \{ y_n \log \sigma(b + \mathbf{w}^{T} \mathbf{x}_n) + (1 - y_n) \log[1 - \sigma(b + \mathbf{w}^{T} \mathbf{x}_n)] \}$$