CSCI567 Machine Learning (Fall 2017)

Prof. Fei Sha

U of Southern California

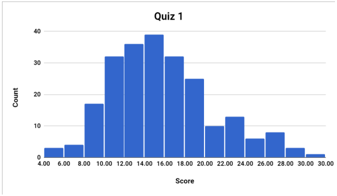
Lecture on Oct. 12, 2017

Outline

Outline

Quiz 1 Preliminary Picture

- Quiz 1 is almost done - we are finishing all but a dozen left
- Basic statistics (out of 31 points without extra credits):
  - Mean: 15.48 (std: 5.12)
  - Median: 15
  - Min: 4
  - Max: 30



Preliminary guideline

- 40% Homework, 60% Quizzes (20% each)
  - If you get 90 out of 100 on all the HW, then you get 36 points
  - If you get 50 out of 100 on Quiz 1 (ie, 15 out of 31), you will earn 10 points towards the final grade
  - If you get 70 out of 100 on Quiz 2, you will get 14 points towards the final grade
  - If you get 80 out of 100 on Quiz 3, you will get 16 points the final grade
  - Total: $36+10+14+16 = 76$

  That is about in the range of [B+ to A-]. To get A, you have to earn 86 and plus.
- For someone continue to get 50% out of every quiz, then you will get into the bracket from B to B+ (for a score of $66 = 36+10+10+10$)

Overall, this Quiz seems to be properly calibrating the class

Outline

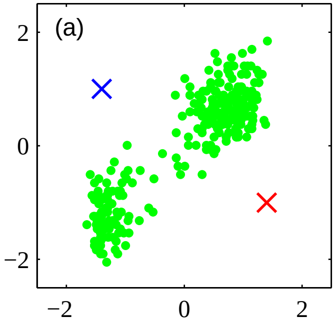
1. Administration
2. Review of last lecture
   - Clustering
   - Gaussian mixture models
   - Demo of GMMs
   - EM Algorithm
   - Relation between K-means and GMMs
3. Naive Bayes

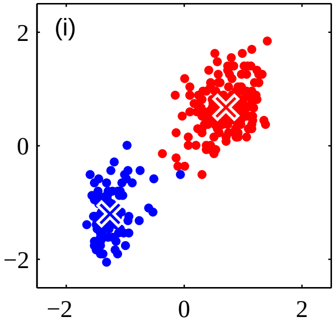
Clustering

Setup Given $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$ and K, we want to output

- $\{\boldsymbol{\mu}_k\}_{k=1}^K$: prototypes of clusters
- $A(\boldsymbol{x}_n) \in \{1, 2, \ldots, K\}$: the cluster membership, i.e., the cluster ID assigned to $\boldsymbol{x}_n$

Example Cluster data into two clusters.





Algorithm: K-means clustering

Intuition Data points assigned to cluster k should be close to $\boldsymbol{\mu}_k$, the prototype.

Distortion measure (clustering objective function, cost function)

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|_2^2$$

where $r_{nk} \in \{0, 1\}$ is an indicator variable

$$r_{nk} = 1 \quad \text{if and only if} \quad A(\boldsymbol{x}_n) = k$$

Algorithm

Minimize distortion measure alternative optimization between $\{r_{nk}\}$ and $\{\boldsymbol{\mu}_k\}$

- **Step 0** Initialize $\{\boldsymbol{\mu}_k\}$ to some values
- **Step 1** Assume the current value of $\{\boldsymbol{\mu}_k\}$ fixed, minimize J over $\{r_{nk}\}$, which leads to the following cluster assignment rule

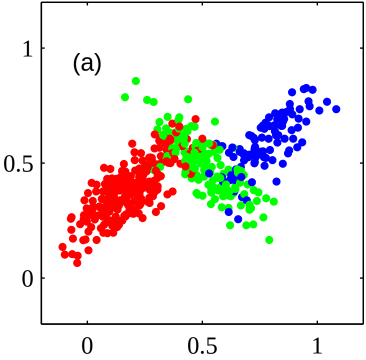
$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

- **Step 2** Assume the current value of $\{r_{nk}\}$ fixed, minimize J over $\{\boldsymbol{\mu}_k\}$, which leads to the following rule to update the prototypes of the clusters

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\boldsymbol{x}_n}{\sum_n r_{nk}}$$

- **Step 3** Determine whether to stop or return to Step 1

Gaussian mixture models: intuition



We will model each region with a Gaussian distribution. This leads to the idea of Gaussian mixture models (GMMs) or mixture of Gaussians (MoGs).

challenge: i) we do not know which (color) region a data point comes from; ii) the parameters of Gaussian distributions in each region. We need to find all of them from *unsupervised* data $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^{N}$.

Gaussian mixture models: formal definition

A Gaussian mixture model has the following density function for $\boldsymbol{x}$

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \omega_k N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \sum_{k=1}^{K} \omega_k \frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}_k|}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^{\mathrm{T}} \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x}-\boldsymbol{\mu}_k)}$$

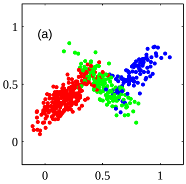
where

- K: the number of Gaussians — they are called (mixture) components
- $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$: mean and covariance matrix of the k-th component
- ω_k: mixture weights – they represent how much each component contributes to the final distribution. It satisfies two properties:

$$\forall\ k,\ \omega_k > 0, \quad \text{and} \quad \sum_k \omega_k = 1$$

The properties ensure $p(\boldsymbol{x})$ is a properly normalized probability density function.

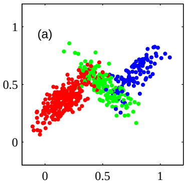
GMM as the marginal distribution of a joint distribution: example



The conditional distribution between $\boldsymbol{x}$ and z (representing color) are

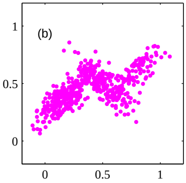
$$p(\boldsymbol{x}|z =' red') = \color{red}{N(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)}$$
$$p(\boldsymbol{x}|z =' blue') = \color{blue}{N(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)}$$
$$p(\boldsymbol{x}|z =' green') = \color{green}{N(\boldsymbol{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)}$$

GMM as the marginal distribution of a joint distribution: example



The conditional distribution between $\boldsymbol{x}$ and z (representing color) are

$$p(\boldsymbol{x}|z='red') = N(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$$
$$p(\boldsymbol{x}|z='blue') = N(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$
$$p(\boldsymbol{x}|z='green') = N(\boldsymbol{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$



The marginal distribution is thus

$$p(\boldsymbol{x}) = p('red')N(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + p('blue')N(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) + p('green')N(\boldsymbol{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$

Parameter estimation for Gaussian mixture models

The parameters in GMMs are $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$. To estimate, consider the simple case first.

z **is given** If we assume z is observed for every $\boldsymbol{x}$, then our estimation problem is easier to solve. Particularly, our training data is *augmented*

$$\mathcal{D}' = \{\boldsymbol{x}_n, z_n\}_{n=1}^{N}$$

Note that, for every $\boldsymbol{x}_n$, we have a z_n to denote the region/color where the specific $\boldsymbol{x}_n$ comes from. We call $\mathcal{D}'$ the *complete* data and $\mathcal{D}$ the *incomplete* data.

Parameter estimation for Gaussian mixture models

The parameters in GMMs are $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$. To estimate, consider the simple case first.

z **is given** If we assume z is observed for every $\boldsymbol{x}$, then our estimation problem is easier to solve. Particularly, our training data is *augmented*

$$\mathcal{D}' = \{\boldsymbol{x}_n, z_n\}_{n=1}^{N}$$

Note that, for every $\boldsymbol{x}_n$, we have a z_n to denote the region/color where the specific $\boldsymbol{x}_n$ comes from. We call $\mathcal{D}'$ the *complete* data and $\mathcal{D}$ the *incomplete* data.

Given $\mathcal{D}'$, the maximum likelihood estimation of the $\boldsymbol{\theta}$ is given by

$$\boldsymbol{\theta} = \arg\max \log \mathcal{D}' = \sum_n \log p(\boldsymbol{x}_n, z_n)$$

Key points for finding solution for complete data

Likelihood is decompsed so we can estimate different components separately

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n \gamma_{nk} \log \omega_k + \sum_k \left\{ \sum_n \gamma_{nk} \log N(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

where γ_{nk} is 1 if $z_n = k$. Note that, the term inside the braces depends on k-th component's parameters. It is now easy to show that (left as a homework exercise), the maximum likelihood estimation of the parameters are

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}$$

Intuition

Since γ_{nk} is binary, the previous solution is nothing but

- For ω_k: count the number of data points whose z_n is k and divide by the total number of data points (note that $\sum_k \sum_n \gamma_{nk} = N$)
- For $\boldsymbol{\mu}_k$: get all the data points whose z_n is k, compute their mean
- For $\boldsymbol{\Sigma}_k$: get all the data points whose z_n is k, compute their covariance matrix

This intuition is going to help us to develop an algorithm for estimating $\boldsymbol{\theta}$ when we do not know z_n.

Parameter estimation for GMMs: complete vs. incomplete data

Complete Data

γ_{nk} is binary as z_n is given

$$\gamma_{nk} = \mathbb{I}[z_n = k]$$

Incomplete Data

γ_{nk} is "guessed" as z_n is not given

$$p(z_n = k|\boldsymbol{x}_n) = \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{p(\boldsymbol{x}_n)} \tag{1}$$

$$= \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{\sum_{k'=1}^{K} p(\boldsymbol{x}_n|z_n = k')p(z_n = k')} \tag{2}$$

Same estimation formula

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}$$

How to compute γ_{nk}?

$$p(z_n = k|\boldsymbol{x}_n) = \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{p(\boldsymbol{x}_n)} \tag{3}$$

$$= \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{\sum_{k'=1}^{K} p(\boldsymbol{x}_n|z_n = k')p(z_n = k')} \tag{4}$$

Note that, to compute the posterior probability, we need to know the parameters $\boldsymbol{\theta}$.

This implies an *iterative* procedure.

Iterative procedure

Since we do not know $\boldsymbol{\theta}$ to begin with, we cannot compute the soft γ_{nk}. However, we can invoke an iterative procedure and alternate between estimating γ_{nk} and using the estimated γ_{nk} to compute the parameters

- Step 0: guess $\boldsymbol{\theta}$ with initial values
- Step 1: compute γ_{nk} using the current $\boldsymbol{\theta}$
- Step 2: update $\boldsymbol{\theta}$ using the just computed γ_{nk}
- Step 3: go back to Step 1

Questions: i) is this procedure correct, for example, optimizing a sensible criteria? ii) practically, will this procedure ever stop instead of iterating forever?

The answer lies in the EM algorithm — a powerful procedure for model estimation with unknown data.

Here we show how EM works

EM algorithm: motivation and setup

As a general procedure, EM is used to estimate parameters for probabilistic models with hidden/latent variables. Suppose the model is given by a joint distribution

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})$$

where $\boldsymbol{x}$ is the observed random variable and $\boldsymbol{z}$ is hidden.

We are given data containing only the observed variable $\mathcal{D} = \{\boldsymbol{x}_n\}$ where the corresponding hidden variable values $\boldsymbol{z}$ is not included. Our goal is to obtain the maximum likelihood estimate of $\boldsymbol{\theta}$. Namely, we choose

$$\begin{aligned}\boldsymbol{\theta} &= \arg\max \log \mathcal{D} = \arg\max \sum_n \log p(\boldsymbol{x}_n|\boldsymbol{\theta}) \\ &= \arg\max \sum_n \log \sum_{\boldsymbol{z}_n} p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta})\end{aligned}$$

The objective function $\ell(\boldsymbol{\theta})$ is called *incomplete* log-likelihood.

Expected (complete) log-likelihood

The difficulty with incomplete log-likelihood is that it needs to sum over all possible values that z_n can take, then take a logarithm. This log-sum format makes computation intractable. Instead, the EM algorithm uses a clever trick to change this into sum-log form.

To this end, we define the following

$$\begin{aligned} Q_q(\boldsymbol{\theta}) &= \sum_n \mathbb{E}_{\boldsymbol{z}_n \sim q(\boldsymbol{z}_n)} \log p(\boldsymbol{x}_n, \boldsymbol{z}_n | \boldsymbol{\theta}) \\ &= \sum_n \sum_{\boldsymbol{z}_n} q(\boldsymbol{z}_n) \log p(\boldsymbol{x}_n, \boldsymbol{z}_n | \boldsymbol{\theta}) \end{aligned}$$

which is called *expected (complete) log-likelihood* (with respect to $q(\boldsymbol{z})$. $q(\boldsymbol{z})$ is a distribution over z. Note that $Q_q(\boldsymbol{\theta})$ takes the form of sum-log, which turns out to be tractable.

Examples

Consider the previous model where $\boldsymbol{x}$ could be from 3 regions. We can choose $q(\boldsymbol{z})$ any valid distribution. This will lead to different $Q_q(\boldsymbol{\theta})$. Note that z here represents different colors.

- $q(z = k) = 1/3$ for any of 3 colors. This gives rise to

$$Q_q(\boldsymbol{\theta}) = \sum_n \frac{1}{3} \left[\log p(\boldsymbol{x}_n, 'red'|\boldsymbol{\theta}) \right. \\ \left. + \log p(\boldsymbol{x}_n, 'blue'|\boldsymbol{\theta}) + \log p(\boldsymbol{x}_n, 'green'|\boldsymbol{\theta})\right]$$

- $q(z = k) = 1/2$ for 'red' and 'blue', 0 for 'green'. This gives rise to

$$Q_q(\boldsymbol{\theta}) = \sum_n \frac{1}{2} \left[\log p(\boldsymbol{x}_n, 'red'|\boldsymbol{\theta}) + \log p(\boldsymbol{x}_n, 'blue'|\boldsymbol{\theta})\right]$$

Which $q(\boldsymbol{z})$ to choose?

We will choose a special $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})$, i.e., the posterior probability of $\boldsymbol{z}$. We define

$$Q(\boldsymbol{\theta}) = Q_{\boldsymbol{z}\sim p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})}(\boldsymbol{\theta})$$

and we will show

$$\ell(\boldsymbol{\theta}) = Q(\boldsymbol{\theta}) + \sum_n \mathbb{H}[p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta})]$$

where $\mathbb{H}[p]$ is the entropy of the probabilistic distribution p:

$$\mathbb{H}[p(\boldsymbol{x})] = -\int p(\boldsymbol{x}) \log p(\boldsymbol{x}) d\boldsymbol{x}$$

Proof (not required to memorize)

$$
\begin{aligned}
Q(\boldsymbol{\theta}) &= \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}) \log p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta}) \\
&= \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}) \left[\log p(\boldsymbol{x}_n|\boldsymbol{\theta}) + \log p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta})\right] \\
&= \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}) \log p(\boldsymbol{x}_n|\boldsymbol{\theta}) \\
&\qquad + \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}) \log p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}) \\
&= \sum_n \log p(\boldsymbol{x}_n|\boldsymbol{\theta}) \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}) - \sum_n \mathbb{H}[p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta})] \\
&= \sum_n \log p(\boldsymbol{x}_n|\boldsymbol{\theta}) - \sum_n \mathbb{H}[p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta})] \\
&= \ell(\boldsymbol{\theta}) - \sum_n \mathbb{H}[p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta})]
\end{aligned}
$$

A computable $Q(\boldsymbol{\theta})$

As before, $Q(\boldsymbol{\theta})$ cannot be computed, as it depends on the unknown parameter values $\boldsymbol{\theta}$ to compute the posterior probability $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})$. Instead, we will use a known value $\boldsymbol{\theta}^{\mathrm{OLD}}$ to compute the expected likelihood

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{OLD}}) = \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}^{\mathrm{OLD}}) \log p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta})$$

Note that, in the above, the variable is $\boldsymbol{\theta}$. $\boldsymbol{\theta}^{\mathrm{OLD}}$ is assumed to be known. By its definition, the following is true

$$Q(\boldsymbol{\theta}) = Q(\boldsymbol{\theta}, \boldsymbol{\theta})$$

However, how does $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{OLD}})$ relates to $\ell(\boldsymbol{\theta})$? We will show that

$$\ell(\boldsymbol{\theta}) \geq Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{OLD}}) + \sum_n \mathbb{H}[p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta}^{\mathrm{OLD}})]$$

Thus, in a way, $Q(\boldsymbol{\theta})$ is better than $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{OLD}})$ (because we have equality there) except that we cannot compute the former.

Proof (not required to memorize)

$$\begin{aligned}
\ell(\boldsymbol{\theta}) &= \sum_n \log \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta}^{\text{OLD}}) \frac{p(\boldsymbol{x}_n,\boldsymbol{z}_n|\boldsymbol{\theta})}{p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta}^{\text{OLD}})} \\
&\geq \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta}^{\text{OLD}}) \log \frac{p(\boldsymbol{x}_n,\boldsymbol{z}_n|\boldsymbol{\theta})}{p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta}^{\text{OLD}})} \\
&= \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta}^{\text{OLD}}) \log p(\boldsymbol{x}_n,\boldsymbol{z}_n|\boldsymbol{\theta}) \\
&\qquad - \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta}^{\text{OLD}}) \log p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta}^{\text{OLD}}) \\
&= Q(\boldsymbol{\theta},\boldsymbol{\theta}^{\text{OLD}}) + \sum_n \mathbb{H}[p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta}^{\text{OLD}})]
\end{aligned}$$

The inequality ($\geq$) is true because $\log$ is a concave function:

$$\log \sum_i w_i x_i \geq \sum_i w_i \log x_i, \quad \forall\ w_i \geq 0, \quad \sum_i w_i = 1$$

And in our case, the w_i is $p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta}^{\text{OLD}})$.

Putting things together: auxiliary function

So far we have shown a lower bound on the log-likelihood

$$\ell(\boldsymbol{\theta}) \geq A(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{OLD}}) = Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{OLD}}) + \sum_n \mathbb{H}[p(\boldsymbol{z}|\boldsymbol{x}_n; \boldsymbol{\theta}^{\mathrm{OLD}})]$$

We will call the right-hand-side an *auxiliary function*.

This auxiliary function has an important property. When $\boldsymbol{\theta} = \boldsymbol{\theta}^{\mathrm{OLD}}$,

$$A(\boldsymbol{\theta}, \boldsymbol{\theta}) = \ell(\boldsymbol{\theta})$$

Use auxiliary function to increase log-likelihood

Suppose we have an initial guess $\boldsymbol{\theta}^{\mathrm{OLD}}$, then we maximize the *auxiliary function*

$$\boldsymbol{\theta}^{\mathrm{NEW}} = \arg\max_{\boldsymbol{\theta}} A(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{OLD}})$$

Use auxiliary function to increase log-likelihood

Suppose we have an initial guess $\boldsymbol{\theta}^{\text{OLD}}$, then we maximize the *auxiliary function*

$$\boldsymbol{\theta}^{\text{NEW}} = \arg\max_{\boldsymbol{\theta}} A(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{OLD}})$$

With the new guess, we have

$$\ell(\boldsymbol{\theta}^{\text{NEW}}) \geq A(\boldsymbol{\theta}^{\text{NEW}}, \boldsymbol{\theta}^{\text{OLD}}) \geq A(\boldsymbol{\theta}^{\text{OLD}}, \boldsymbol{\theta}^{\text{OLD}}) = \ell(\boldsymbol{\theta}^{\text{OLD}})$$

Use auxiliary function to increase log-likelihood

Suppose we have an initial guess $\boldsymbol{\theta}^{\mathrm{OLD}}$, then we maximize the *auxiliary function*

$$\boldsymbol{\theta}^{\mathrm{NEW}} = \arg\max_{\boldsymbol{\theta}} A(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{OLD}})$$

With the new guess, we have

$$\ell(\boldsymbol{\theta}^{\mathrm{NEW}}) \geq A(\boldsymbol{\theta}^{\mathrm{NEW}}, \boldsymbol{\theta}^{\mathrm{OLD}}) \geq A(\boldsymbol{\theta}^{\mathrm{OLD}}, \boldsymbol{\theta}^{\mathrm{OLD}}) = \ell(\boldsymbol{\theta}^{\mathrm{OLD}})$$

Repeating this process, we have

$$\ell(\boldsymbol{\theta}^{\mathrm{EVEN\ NEWER}}) \geq \ell(\boldsymbol{\theta}^{\mathrm{NEW}}) \geq \ell(\boldsymbol{\theta}^{\mathrm{OLD}})$$

where

$$\boldsymbol{\theta}^{\mathrm{EVEN\ NEWER}} = \arg\max_{\boldsymbol{\theta}} A(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{NEW}})$$

Iterative and monotonic improvement

Thus, by maximizing the auxiliary function, we obtain a sequence of guesses

$$\boldsymbol{\theta}^{\text{OLD}}, \boldsymbol{\theta}^{\text{NEW}}, \boldsymbol{\theta}^{\text{EVEN NEWER}}, \cdots,$$

that will keep increasing the likelihood. This process will eventually stops if the likelihood is bounded from above (i.e.. less than $+\infty$). This is the core of the EM algorithm.

Expectation-Maximization (EM)

- Step 0: Initialize $\boldsymbol{\theta}$ with $\boldsymbol{\theta}^{(0)}$
- Step 1 (E-step): Compute the auxiliary function using the current value of $\boldsymbol{\theta}$

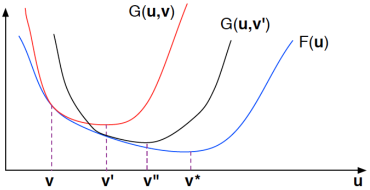
$$A(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})$$

- Step 2 (M-step): Maximize the auxiliary function

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \arg\max A(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})$$

- Step 3: Increase t to $t+1$ and go back to Step 1; or stop if $\ell(\boldsymbol{\theta}^{(t+1)})$ does not improve $\ell(\boldsymbol{\theta}^{(t)})$ much.

Auxiliary function (for minimizing)



- Target: minimize $F(u)$
- Auxiliary: $G(u,v) \geq F(u)$ and $G(u,u) = F(u)$
- Sequence of improvement

$$v' = \arg\min G(u,v) \rightarrow v'' = \arg\min G(u,v')$$
$$\rightarrow v''' = \arg\min G(u,v'') \cdots$$
$$F(v) \geq F(v') \geq F(v'') \geq \cdots$$

Auxiliary function used in EM

For the incomplete likelihood $\ell(\theta)$,

$$\ell(\boldsymbol{\theta}) \geq A(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{OLD}}) = Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{OLD}}) + \sum_n \mathbb{H}[p(\boldsymbol{z}|\boldsymbol{x}_n; \boldsymbol{\theta}^{\text{OLD}})]$$

where the expected likelihood

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{OLD}}) = \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n; \boldsymbol{\theta}^{\text{OLD}}) \log p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta})$$

Remarks

- The EM procedure converges but only converges to a local optimum. Global optimum is not guaranteed to be found.
- The E-step depends on computing the posterior probability

$$p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}^{(t)})$$

- The M-step does not depend on the entropy term, so we need only to do the following

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \arg\max A(\boldsymbol{\theta},\boldsymbol{\theta}^{(t)}) = \arg\max Q(\boldsymbol{\theta},\boldsymbol{\theta}^{(t)})$$

We often call the last term Q-function.

Example: applying EM to GMMs

What is the E-step in GMM? We compute the responsibility

$$\gamma_{nk} = p(z = k|\boldsymbol{x}_n; \boldsymbol{\theta}^{(t)})$$

What is the M-step in GMM? The Q-function is

$$\begin{aligned}
Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) &= \sum_n \sum_k p(z = k|\boldsymbol{x}_n; \boldsymbol{\theta}^{(t)}) \log p(\boldsymbol{x}_n, z = k|\boldsymbol{\theta}) \\
&= \sum_n \sum_k \gamma_{nk} \log p(\boldsymbol{x}_n, z = k|\boldsymbol{\theta}) \\
&= \sum_k \sum_n \gamma_{nk} \log p(z = k) p(\boldsymbol{x}_n|z = k) \\
&= \sum_k \sum_n \gamma_{nk} \left[\log \omega_k + \log N(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right]
\end{aligned}$$

Hence, we have recovered the parameter estimation algorithm for GMMs, seen previously. (We still need to do the maximization to get $\boldsymbol{\theta}^{(t+1)}$ — left as homework.)

GMMs and K-means

GMMs provide probabilistic interpretation for K-means. We have the following observation:

- Assume all Gaussian components have $\sigma^2 \boldsymbol{I}$ as their covariance matrices
- Further assume $\sigma \to 0$
- Thus, we only need to estimate $\boldsymbol{\mu}_k$, i.e., means
- Then, the EM for GMM parameter estimation simplifies to K-means.

For this reason, K-means is often called "hard" GMM or GMMs is called "soft" K-means. The soft posterior γ_{nk} provides a probabilistic assignment for $\boldsymbol{x}_n$ to cluster k represented by the corresponding Gaussian distribution.

Outline

Unsupervised learning

So far we have described how to model data is distributed

Given $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, . . . , $\boldsymbol{x}_N$, what is the possible model for

$$p(\boldsymbol{x})$$

Unsupervised learning

So far we have described how to model data is distributed

Given $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, . . . , $\boldsymbol{x}_N$, what is the possible model for

$$p(\boldsymbol{x})$$

We can also ask

Given $(\boldsymbol{x}_1, y_1)$, $(\boldsymbol{x}_2, y_2)$, . . . , $(\boldsymbol{x}_N, y_N)$, what is the possible model for

$$p(\boldsymbol{x}, y)$$

We will see that if we know $p(\boldsymbol{x}, y)$, we can get an optimal classifier.

Our approach

There are many ways, we will leverage

$$p(\boldsymbol{x}, y) = p(y)p(\boldsymbol{x}|y)$$

to model each part separately.

A daily battle

Great news: I will be rich!

FROM THE DESK OF MR. AMINU SALEH
DIRECTOR, FOREIGN OPERATIONS DEPARTMENT
AFRI BANK PLC
Afribank Plaza,
14th Floormoney344.jpg
51/55 Broad Street,
P.M.B 12021 Lagos-Nigeria

Attention: Honorable Beneficiary,

IMMEDIATE PAYMENT NOTIFICATION

It is my modest obligation to write you th … owed payment through our most respected
financial institution (AFRI BANK PLC). I a… tions Department, AFRI Bank Plc, NIGERIA.
The British Government, in conjunction w… NITED NATIONS ORGANIZATION on
foreign payment matters, has empowered … tion, to handle all foreign payments and
release them to their appropriate benefici… deral Reserve Bank.

To facilitate the process of this transactio… tion below:

1) Your full Name and Address:
2) Phones, Fax and Mobile No. :
3) Profession, Age and Marital Status:
4) Copy of any valid form of your Identification:

How to tell spam from ham?

FROM THE DESK OF MR. AMINU SALEH
DIRECTOR, FOREIGN OPERATIONS DEPARTMENT
AFRI BANK PLC
Afribank Plaza,
14th Floormoney344.jpg
51/55 Broad Street,
P.M.B 12021 Lagos-Nigeria

Attention: Honorable Beneficiary,

IMMEDIATE PAYMENT NOTIFICATION VALUED AT **US$10 MILLION**

Dear Dr.Sha,

I just would like to remind you of your scheduled presentation for CS597, Monday October 13, 12pm at OHE122.

If there is anything that you would need, please do not hesitate to contact me.

sincerely,

Christian Siagian

Intuition

How human solves the problem?

Spam emails

concentrated use of a lot of words like "money", "free", "bank account", "viagara"

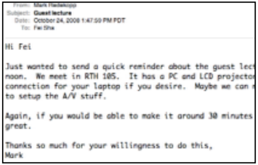
Ham emails

word usage pattern is more spread out

Simple strategy: count the words

Bag-of-word representation of documents (and textual data)

$$\begin{pmatrix} \text{free} & 100 \\ \text{money} & 2 \\ \vdots & \vdots \\ \text{account} & 2 \\ \vdots & \vdots \end{pmatrix}$$


From: Mark Hadekopp
Subject: **Guest lecture**
Date: October 24, 2008 1:47:50 PM PDT
To: Fei Sha

Hi Fei

Just wanted to send a quick reminder about the guest lect
noon. We meet in RTH 105. It has a PC and LCD projector
connection for your laptop if you desire. Maybe we can
to setup the A/V stuff.

Again, if you would be able to make it around 30 minutes
great.

Thanks so much for your willingness to do this,
Mark

$$\begin{pmatrix} \text{free} & 1 \\ \text{money} & 1 \\ \vdots & \vdots \\ \text{account} & 2 \\ \vdots & \vdots \end{pmatrix}$$

Weighted sum of those telltale words

different weights for spam and ham: representing how compatible the word usage pattern is to different category

$$\begin{pmatrix} \text{free} & 100 \\ \text{money} & 2 \\ \vdots & \vdots \\ \text{account} & 2 \\ \vdots & \vdots \end{pmatrix}$$

$$\begin{pmatrix} 100 \times 0.2 \\ 2 \times 0.3 \\ \vdots \\ 2 \times 0.3 \\ \vdots \end{pmatrix} = 3.2$$

$$\begin{pmatrix} 100 \times 0.01 \\ 2 \times 0.02 \\ \vdots \\ 2 \times 0.01 \\ \vdots \end{pmatrix} = 1.03$$

Our intuitive model of classification

Assign weight to each word

Compute compatibility score to "spam"

of "free" x a_{free} + # of "account" x $a_{account}$ + # of "money" x a_{money}

Compute compatibility score to "ham":

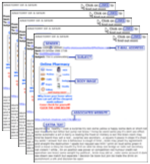
of "free" x b_{free} + # of "account" x $b_{account}$ + # of "money" x b_{money}

Make a decision:

if spam score > ham score then spam

else ham

How we get the weights?

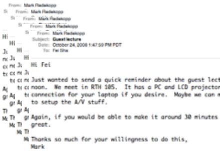


Learning from experience

get a lot of spams

get a lot of hams

But what to optimize?



A probabilistic modeling perspective

Naive Bayes model for identifying spams

Class label: binary

y = {spam, ham}

Features: word counts in the document (Bag-of-word)

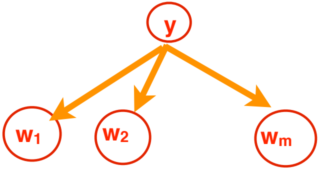
Ex: x = {('free', 100), ('lottery', 10), ('money', 10), , ('identification', 1)...}

Each pair is in the format of
(w_i, #w_i), namely, a unique word in the dictionary,
and the number of times it shows up

Naive Bayes model for identifying spams

$$
\begin{aligned}
p(x|y) &= p(w_1|y)^{\#w_1} p(w_2|y)^{\#w_2} \cdots p(w_m|y)^{\#w_m} \\
&= \prod_i p(w_i|y)^{\#w_i}
\end{aligned}
$$

These conditional probabilities
are model parameters



Spam writer's vocabulary

Features: word counts in the document

Ex: x = {('free', 100), ('identification', 2), ('lottery', 10), ('money', 10), ...}

Model: Naive Bayes (NB)

$$
\begin{aligned}
p(x|\text{spam}) = p(\text{'free'}|\text{spam})^{100} p(\text{'identification'}|\text{spam})^{2} \\
p(\text{'lottery'}|\text{spam})^{10} p(\text{'money'}|\text{spam})^{10} \cdots \\
\neq p(x|\text{ham})
\end{aligned}
$$

Parameters to be estimated:
p('free'|spam), p('free'|ham),etc

Naive Bayes

Why the name "naive"?

Strong assumption of conditional independence:

$$p(w_i, w_j | y) = p(w_i | y) p(w_j | y)$$

How to estimate model parameters?

Use maximum likelihood estimation (soon)

Does this correspond to our intuitive model of classification?

Yes. It does!

Let us consider the Bayes optimal classifier under this assumed probabilistic distribution

$$
\begin{aligned}
p(x|y) &= p(w_1|y)^{\#w_1} p(w_2|y)^{\#w_2} \cdots p(w_m|y)^{\#w_m} \\
&= \prod_i p(w_i|y)^{\#w_i}
\end{aligned}
$$

Bayes optimal classifier

Consider the following classifier, using the posterior probability $\eta(\boldsymbol{x}) = p(y=1|\boldsymbol{x})$

$$f^*(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \eta(\boldsymbol{x}) \geq 1/2 \\ 0 & \text{if } \eta(\boldsymbol{x}) < 1/2 \end{cases} \quad \text{equivalently } f^*(\boldsymbol{x}) = \begin{cases} 1 & \text{if } p(y=1|\boldsymbol{x}) \geq p(y=0|\boldsymbol{x}) \\ 0 & \text{if } p(y=1|\boldsymbol{x}) < p(y=0|\boldsymbol{x}) \end{cases}$$

Bayes optimal classifier

Consider the following classifier, using the posterior probability $\eta(\boldsymbol{x}) = p(y = 1|\boldsymbol{x})$

$$f^*(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \eta(\boldsymbol{x}) \geq 1/2 \\ 0 & \text{if } \eta(\boldsymbol{x}) < 1/2 \end{cases} \quad \text{equivalently } f^*(\boldsymbol{x}) = \begin{cases} 1 & \text{if } p(y=1|\boldsymbol{x}) \geq p(y=0|\boldsymbol{x}) \\ 0 & \text{if } p(y=1|\boldsymbol{x}) < p(y=0|\boldsymbol{x}) \end{cases}$$

Theorem

For any labeling function $f(\cdot)$, $R(f^*, \boldsymbol{x}) \leq R(f, \boldsymbol{x})$ *where* $R(\cdot)$ *is the 0/1 expected risk/loss function. Similarly,* $R(f^*) \leq R(f)$*. Namely,* $f^*(\cdot)$ *is optimal.*

Naive Bayes classification rule

For any document x, we need to compute

$$p(\text{spam}|x) \quad \text{and} \quad p(\text{ham}|x)$$

Naive Bayes classification rule

For any document x, we need to compute

$$p(\text{spam}|x) \quad \text{and} \quad p(\text{ham}|x)$$

Using Bayes rule, this gives rise to

$$p(\text{spam}|x) = \frac{p(x|\text{spam})p(\text{spam})}{p(x)}, \quad p(\text{ham}|x) = \frac{p(x|\text{ham})p(\text{ham})}{p(x)}$$

Naive Bayes classification rule

For any document x, we need to compute

$$p(\text{spam}|x) \quad \text{and} \quad p(\text{ham}|x)$$

Using Bayes rule, this gives rise to

$$p(\text{spam}|x) = \frac{p(x|\text{spam})p(\text{spam})}{p(x)}, \quad p(\text{ham}|x) = \frac{p(x|\text{ham})p(\text{ham})}{p(x)}$$

It is convenient to compute the logarithms, so we need only to compare

$$\log[p(x|\text{spam})p(\text{spam})] \quad \text{versus} \quad \log[p(x|\text{ham})p(\text{ham})]$$

as the denominators are the same

Classifier in the linear form of compatibility scores

$$\log[p(x|\text{spam})p(\text{spam})] = \log\left[\prod_i p(w_i|\text{spam})^{\#w_i} p(\text{spam})\right] \tag{5}$$

$$= \sum_i \#w_i \log p(w_i|\text{spam}) + \log p(\text{spam}) \tag{6}$$

Classifier in the linear form of compatibility scores

$$\log[p(x|\textsf{spam})p(\textsf{spam})] = \log\left[\prod_i p(w_i|\textsf{spam})^{\#w_i} p(\textsf{spam})\right] \tag{5}$$

$$= \sum_i \#w_i \log p(w_i|\textsf{spam}) + \log p(\textsf{spam}) \tag{6}$$

Similarly, we have

$$\log[p(x|\textsf{ham})p(\textsf{ham})] = \sum_i \#w_i \log p(w_i|\textsf{ham}) + \log p(\textsf{ham})$$

Namely, we are back to the idea of comparing weighted sum of # of word occurrences!

$\log p(\textit{spam})$ *and* $\log p(\textit{ham})$ *are called "priors" or "bias" (they are not in our intuition but they are crucially needed)*

Mini-summary

What we have shown

By making a probabilistic model (i.e., Naive Bayes), we are able to derive a decision rule that is consistent with our intuition

Our next step is to leverage this link to learn the rule from the data

Formal definition of Naive Bayes

General case

Given a random variable $X \in \mathbb{R}^{\mathsf{D}}$ and a dependent variable $Y \in [\mathsf{C}]$, the Naive Bayes model defines the joint distribution

$$P(X = x, Y = y) = P(Y = y)P(X = x|Y = y) \tag{7}$$

$$= P(Y = y)\prod_{d=1}^{\mathsf{D}} P(X_d = x_d|Y = y) \tag{8}$$

Special case (i.e., our model of spam emails)

Assumptions

- All X_d are categorical variables from the same domain — $x_d \in [\mathsf{K}]$, 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

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

Goal

Learn $\pi_c, c = 1, 2, \cdots, \mathsf{C}$, and $\theta_{ck}, \forall c \in [\mathsf{C}], k \in [\mathsf{K}]$ 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.

Our hammer: maximum likelihood estimation

Log-Likelihood of the training data

$$\mathcal{L} = \log P(\mathcal{D}) = \log \prod_{n=1}^{\mathsf{N}} \pi_{y_n} P(x_n|y_n) \tag{9}$$

$$= \log \prod_{n=1}^{\mathsf{N}} \left(\pi_{y_n} \prod_k \theta_{y_n k}^{z_{nk}} \right) \tag{10}$$

$$= \sum_n \left(\log \pi_{y_n} + \sum_k z_{nk} \log \theta_{y_n k} \right) \tag{11}$$

$$= \sum_n \log \pi_{y_n} + \sum_{n,k} z_{nk} \log \theta_{y_n k} \tag{12}$$

Our hammer: maximum likelihood estimation

Log-Likelihood of the training data

$$\mathcal{L} = \log P(\mathcal{D}) = \log \prod_{n=1}^{\mathsf{N}} \pi_{y_n} P(x_n | y_n) \tag{9}$$

$$= \log \prod_{n=1}^{\mathsf{N}} \left(\pi_{y_n} \prod_k \theta_{y_n k}^{z_{nk}} \right) \tag{10}$$

$$= \sum_n \left(\log \pi_{y_n} + \sum_k z_{nk} \log \theta_{y_n k} \right) \tag{11}$$

$$= \sum_n \log \pi_{y_n} + \sum_{n,k} z_{nk} \log \theta_{y_n k} \tag{12}$$

Optimize it!

$$(\pi_c^*, \theta_{ck}^*) = \arg\max \sum_n \log \pi_{y_n} + \sum_{n,k} z_{nk} \log \theta_{y_n k}$$

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 $\{\theta_{ck}, k = 1, 2, \cdots, \mathsf{K}\}$ and $\{\theta_{c'k}, k = 1, 2, \cdots, \mathsf{K}\}$ 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 π_c (total C sides)
- And we have total N trials of rolling this dice

Solution

$$\pi_c^* = \frac{\text{\#of data points labeled as c}}{\text{N}}$$

Estimating $\{\theta_{ck}, k = 1, 2, \cdots, \mathsf{K}\}$

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 θ_{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)

$$p(\text{funny\_word}|\text{ham}) = \frac{\#\text{of funny\_word in ham emails}}{\#\text{of words in ham emails}} \tag{13}$$

$$p(\text{funny\_word}|\text{spam}) = \frac{\#\text{of funny\_word in spam emails}}{\#\text{of words in spam emails}} \tag{14}$$

Classification rule

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

$$y^* = \arg\max_{c \in [\mathsf{C}]} P(y = c|x) \quad (15)$$

$$= \arg\max_{c \in [\mathsf{C}]} P(y = c)P(x|y = c) \quad (16)$$

$$= \arg\max_c [\log \pi_c + \sum_k z_k \log \theta_{ck}] \quad (17)$$

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 θ_{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 $\sum_k \theta_{ck} = 1$,

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

Summary

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.

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 is the same as logistic regression's decision boundary. However, we estimate *parameters* differently.

Difference and similarity: can you fill the blank

	Logistic regression	Naive Bayes
Similar	Linear classifier	Linear classifier
Difference	?	?