CSCI-567: Machine Learning (Fall 2019)

Prof. Victor Adamchik

U of Southern California

Nov. 14, 2019

Outline

General EM algorithm

EM is an algorithm to solve MLE with latent variables (not just GMM), i.e. find the maximizer of

$$P(\boldsymbol{\theta}) = \sum_{n=1}^{N} \ln p(\boldsymbol{x}_n \,; \boldsymbol{\theta})$$

Directly solving the objective is intractable. Instead we optimize the lower bound

$$P(\boldsymbol{\theta}) \geq F\left(\boldsymbol{\theta}, \{q_n^{(t)}\}\right)$$

where

$$F\left(\boldsymbol{\theta}, \{q_n^{(t)}\}\right) = \sum_{n=1}^{N}\sum_{k=1}^{K}\left(q_n(k)\ln p(\boldsymbol{x}_n, z_n = k \,; \boldsymbol{\theta}^{(t)}) - q_n(k)\ln q_n(k)\right)$$

General EM algorithm

Step 0 Initialize $\boldsymbol{\theta}^{(1)}$, $t = 1$

Step 1 (E-Step) **update the posterior of latent variables**

$$q_n^{(t)}(z_n = k) = p(z_n = k \mid \boldsymbol{x}_n \,; \boldsymbol{\theta}^{(t)})$$

and obtain **Expectation** of complete likelihood

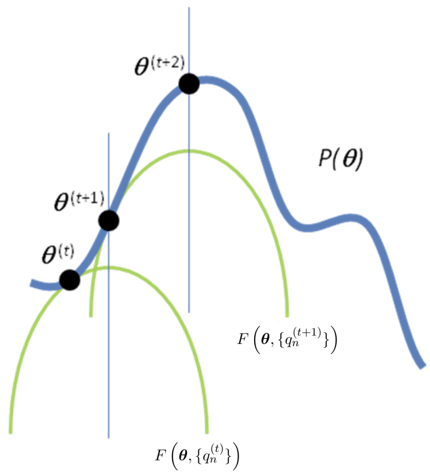
$$Q(\boldsymbol{\theta} \,; \boldsymbol{\theta}^{(t)}) = \sum_{n=1}^{N} \mathbb{E}_{z_n \sim q_n^{(t)}}\left[\ln p(\boldsymbol{x}_n, z_n \,; \boldsymbol{\theta})\right]$$

Step 2 (M-Step) **update the model parameter** via **Maximization**

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \operatorname*{argmax}_{\boldsymbol{\theta}} Q(\boldsymbol{\theta} \,; \boldsymbol{\theta}^{(t)})$$

Step 3 $t \leftarrow t + 1$ and return to Step 1 if not converged

Pictorial explanation



$P(\boldsymbol{\theta})$ is non-concave, but $Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)})$ often is concave and easy to maximize.

$$\begin{aligned} P(\boldsymbol{\theta}^{(t+1)}) &\geq F\left(\boldsymbol{\theta}^{(t+1)} ; \{q_n^{(t)}\}\right) \\ &\geq F\left(\boldsymbol{\theta}^{(t)} ; \{q_n^{(t)}\}\right) \\ &= P(\boldsymbol{\theta}^{(t)}) \end{aligned}$$

So EM always increases the objective value and will converge to some local maximum (similar to K-means).

Apply EM to learn GMMs

E-Step:

$$q_n^{(t)}(z_n = k) = p\left(z_n = k \mid \boldsymbol{x}_n ; \boldsymbol{\theta}^{(t)}\right)$$

This computes the "soft assignment" $\gamma_{nk} = q_n^{(t)}(z_n = k)$, i.e. conditional probability of $\boldsymbol{x}_n$ belonging to cluster k.

Apply EM to learn GMMs

M-Step:

$$\begin{aligned} \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{n=1}^{N} \mathbb{E}_{z_n \sim q_n^{(t)}} \left[\ln p(\boldsymbol{x}_n, z_n ; \boldsymbol{\theta})\right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{n=1}^{N} \mathbb{E}_{z_n \sim q_n^{(t)}} \left[\ln p(z_n ; \boldsymbol{\theta}) + \ln p(\boldsymbol{x}_n | z_n ; \boldsymbol{\theta})\right] \\ &= \underset{\{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}}{\operatorname{argmax}} \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} \left(\ln \omega_k + \ln N(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) \end{aligned}$$

To find $\omega_1, \ldots, \omega_K$, solve

$$\underset{\boldsymbol{\omega}}{\operatorname{argmax}} \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} \ln \omega_k$$

To find each $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$, solve

$$\underset{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k}{\operatorname{argmax}} \sum_{n=1}^{N} \gamma_{nk} \ln N(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

M-Step (continued)

Solutions to previous two problems are very natural, for each k

$$\omega_k = \frac{\sum_n \gamma_{nk}}{N}$$

i.e. (weighted) fraction of examples belonging to cluster k

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

i.e. (weighted) average of examples belonging to cluster k

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

i.e (weighted) covariance of examples belonging to cluster k

GMM: putting it together

EM for clustering:

Step 0 Initialize $\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ for each $k \in [K]$

Step 1 (E-Step) **update the "soft assignment"** (fixing parameters)

$$\gamma_{nk} = p(z_n = k \mid \boldsymbol{x}_n)$$

Step 2 (M-Step) **update the model parameter** (fixing assignments)

$$\omega_k = \frac{\sum_n \gamma_{nk}}{N} \qquad \boldsymbol{\mu}_k = \frac{\sum_n \gamma_{nk}\boldsymbol{x}_n}{\sum_n \gamma_{nk}}$$

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

Step 3 return to Step 1 if not converged

Connection to K-means

K-means is in fact a special case of EM for (a simplified) GMM:

Let $\boldsymbol{\Sigma}_k = \sigma^2 \boldsymbol{I}$ for some fixed σ, so only ω_k and $\boldsymbol{\mu}_k$ are parameters.

EM becomes K-means:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{n=1}^{N} p(\boldsymbol{x}_n\,; \boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{n=1}^{N} \sum_{k=1}^{K} p(z_n = k) N(\boldsymbol{x}_n | \boldsymbol{\mu}_k)$$

If we assume hard assignments $p(z_n = k) = 1$, if $k = C(n)$, then

$$\underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{n=1}^{N} p(\boldsymbol{x}_n\,; \boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{n=1}^{N} N(\boldsymbol{x}_n | \boldsymbol{\mu}_{C(n)})$$

$$= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{n=1}^{N} \exp\left(\frac{-1}{2\sigma^2}\|\boldsymbol{x}_n - \boldsymbol{\mu}_{C(n)}\|_2^2\right) = \underset{\boldsymbol{\mu}, C}{\operatorname{argmin}} \sum_{n=1}^{N} \|\boldsymbol{x}_n - \boldsymbol{\mu}_{C(n)}\|_2^2$$

GMM is a soft version of K-means and it provides a probabilistic interpretation of the data.

Outline

Density estimation

Observe what we have done indirectly for clustering with GMMs is:

Given a training set $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, **estimate a density function** p **that could have generated this dataset** (via $\boldsymbol{x}_n \overset{i.i.d.}{\sim} p$).

We say that a random variable x has a probability distribution $p(x)$.

This is exactly the problem of *density estimation*, another important unsupervised learning problem.

Useful for many downstream applications

- we have seen clustering already, will see more applications today
- these applications also *provide a way to measure quality of the density estimator*

Parametric generative models

Parametric estimation assumes a generative model parametrized by $\boldsymbol{\theta}$:

$$p(\boldsymbol{x}) = p(\boldsymbol{x}\,; \boldsymbol{\theta})$$

here $p(x)$ is a common (predefined) probability distribution. Examples:

- GMM: $p(\boldsymbol{x}\,; \boldsymbol{\theta}) = \sum_{k=1}^{K} \omega_k N(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ where $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$
- Multinomial for 1D examples with K possible values

$$p(x = k\,; \boldsymbol{\theta}) = \theta_k$$

  where $\boldsymbol{\theta}$ is a distribution over K elements.

Size of $\boldsymbol{\theta}$ is independent of the training set size, so it's parametric.

Parametric methods

Again, we apply **MLE** to learn the parameters $\boldsymbol{\theta}$:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmax}} = \sum_{n=1}^{N} \ln p(x_n\,; \boldsymbol{\theta})$$

For some cases this is intractable and we can use EM to approximately solve MLE (e.g. GMMs).

For some other cases this admits a simple closed-form solution (e.g. multinomial).

MLE for multinomial

$$\begin{aligned}\underset{\boldsymbol{\theta}}{\operatorname{argmax}} &= \sum_{n=1}^{N} \ln p(x = x_n\,; \boldsymbol{\theta}) = \sum_{n=1}^{N} \ln \theta_{x_n} \\ &= \sum_{k=1}^{K} \sum_{n: x_n = k} \ln \theta_k = \sum_{k=1}^{K} z_k \ln \theta_k\end{aligned}$$

where $z_k = |\{n : x_n = k\}|$ is **the number of examples with value k**.

The solution (verify yourself!) is simply

$$\theta_k = \frac{z_k}{N} \propto z_k,$$

i.e. the fraction of examples with value k.

Nonparametric models

Can we estimate *without* assuming a fixed generative model?

Kernel density estimation (KDE) is a common approach for nonparametric density estimation (without a pre-defined distribution).

Here "kernel" means something different from what we have seen for "kernel function".

The scikit-learn library provides the KernelDensity class that implements KDE.

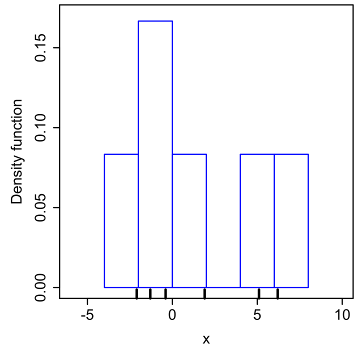
We focus on the 1D (continuous) case.

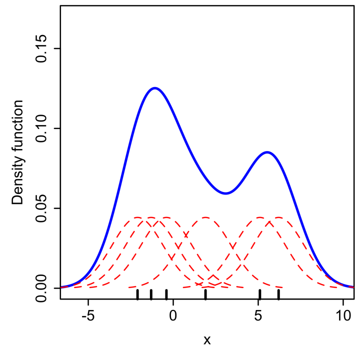
High level idea

picture from Wikipedia

KDE is closely related to a **histogram**. A histogram is a plot that involves first grouping the observations into bins and counting the number of events that fall into each bin. To construct KDE,

- for each data point, create a "hump" (via a kernel)
- sum up all the humps; more data - a higher hump





Kernel

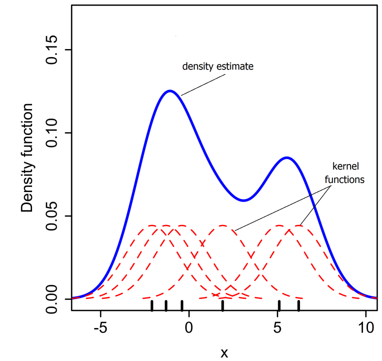
KDE with **a kernel** $K(x)$: $\mathbb{R} \to \mathbb{R}$ centered at x_n:

$$p(x) = \frac{1}{N} \sum_{n=1}^{N} K(x - x_n)$$

Many choices for K, for example, $K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$, the standard Gaussian density

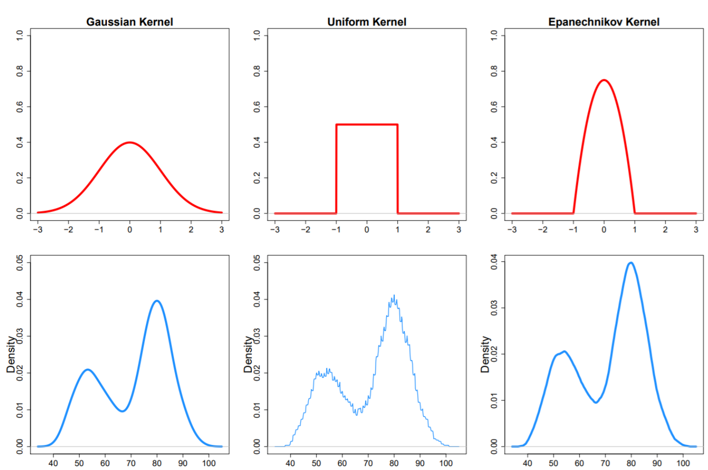
Properties of a kernel:

- symmetry: $K(x) = K(-x)$
- $\int_{-\infty}^{\infty} K(x)dx = 1$, this insures p is a density function.



Different kernels $K(x)$

$$\frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \qquad \frac{1}{2}\mathbb{I}[|x| \leq 1] \qquad \frac{3}{4}\max\{1 - x^2, 0\}$$



Bandwidth

If $K(x)$ is a kernel, then for any $h > 0$

$$K_h(u) \triangleq \frac{1}{h} K\left(\frac{x}{h}\right) \qquad \text{(stretching the kernel)}$$

can be used as a kernel too (verify the two properties yourself)

So, general KDE is determined by both the kernel K and the bandwidth h

$$p(x) = \frac{1}{N} \sum_{n=1}^{N} K_h(x - x_n) = \frac{1}{Nh} \sum_{n=1}^{N} K\left(\frac{x - x_n}{h}\right)$$

- x_n controls the center of each hump
- h controls the width/variance of the humps

Effect of bandwidth

picture from Wikipedia

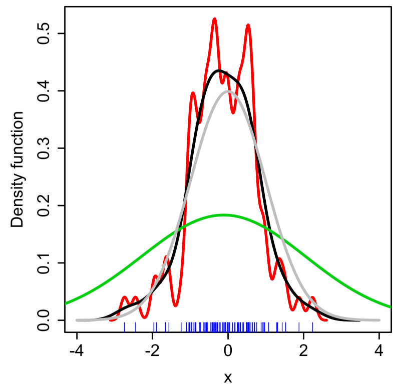
A larger h will smooth a density.

A small h will yield a density that is spiky and very hard to interpret.

Assume Gaussian kernel.

Gray curve is ground-truth

- Red: $h = 0.05$
- Black: $h = 0.337$
- Green: $h = 2$



Bandwidth selection

Selecting h is a deep topic

- one can also do cross-validation based on downstream applications
- there are theoretically-motivated approaches

  Find a value of h that minimizes the error between the estimated density and the true density:

$$\mathbb{E}\left[(p_{KDE}(x) - p(x))^2\right] = \mathbb{E}\left[p_{KDE}(x) - p(x)\right]^2 + Var\left[p_{KDE}(x)\right]$$

  This expression is an example of the bias-variance tradeoff, which we saw in the earlier lecture.

Summary

This was a gentle introduction to probability density estimation.

- Histogram provides a fast and reliable way to visualize the probability density of data.
- Parametric probability density estimation involves selecting a common distribution and estimating the parameters for the density function from data.
- Nonparametric probability density estimation involves using an algorithm (KDE) to fit a model to the arbitrary distribution of data.

Outline

Bayes optimal classifier

Suppose the data $(\boldsymbol{x}_n, y_n)$ is drawn from a joint distribution $p(x, y)$, the **Bayes optimal classifier** is

$$f^*(\boldsymbol{x}) = \underset{c \in [\mathsf{C}]}{\operatorname{argmax}}\, p(c \mid \boldsymbol{x})$$

i.e. predict the class with the largest conditional probability.

$p(x, y)$ is of course unknown, but we can estimate it, which is *exactly a density estimation problem!*

Observe that

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

To estimate $p(\boldsymbol{x} \mid y = c)$ for some $c \in [\mathsf{C}]$, we are doing density estimation using data with label $y = c$.

Discrete features

For a label $c \in [\mathsf{C}]$,

$$p(y = c) = \frac{|\{n : y_n = c\}|}{N}$$

For each possible value k of a discrete feature d,

$$p(x_d = k \mid y = c) = \frac{|\{n : x_{nd} = k, y_n = c\}|}{|\{n : y_n = c\}|}$$

Continuous features

If the feature is continuous, we can do

- parametric estimation, e.g. via a Gaussian

$$p(x_d = x \mid y = c) = \frac{1}{\sqrt{2\pi}\sigma_{cd}} \exp\left(-\frac{(x - \mu_{cd})^2}{2\sigma_{cd}^2}\right)$$

  where μ_{cd} and σ_{cd}^2 are the empirical mean and variance of feature d among all examples with label c.

- or nonparametric estimation, e.g. via a kernel K and bandwidth h:

$$p(x_d = x \mid y = c) = \frac{1}{|\{n : y_n = c\}|} \sum_{n: y_n = c} K_h(x - x_{nd})$$

How to predict?

Using Naive Bayes assumption:

$$p(\boldsymbol{x} \mid y = c) = \prod_{d=1}^{\mathsf{D}} p(x_d \mid y = c)$$

the **prediction** for a new example x is

$$\begin{aligned}
\underset{c \in [\mathsf{C}]}{\operatorname{argmax}}\; p(y = c \mid \boldsymbol{x}) &= \underset{c \in [\mathsf{C}]}{\operatorname{argmax}}\; \frac{p(\boldsymbol{x} \mid y = c)p(y = c)}{p(\boldsymbol{x})} \\
&= \underset{c \in [\mathsf{C}]}{\operatorname{argmax}}\; \left(p(y = c) \prod_{d=1}^{\mathsf{D}} p(x_d \mid y = c)\right) \\
&= \underset{c \in [\mathsf{C}]}{\operatorname{argmax}}\; \left(\ln p(y = c) + \sum_{d=1}^{\mathsf{D}} \ln p(x_d \mid y = c)\right)
\end{aligned}$$

Naive Bayes

For **discrete features**, plugging in previous MLE estimations gives

$$\begin{aligned}
&\underset{c\in[\mathsf{C}]}{\operatorname{argmax}}\ p(y=c \mid x) \\
&= \underset{c\in[\mathsf{C}]}{\operatorname{argmax}} \left(\ln p(y=c) + \sum_{d=1}^{\mathsf{D}} \ln p(x_d \mid y=c) \right) \\
&= \underset{c\in[\mathsf{C}]}{\operatorname{argmax}} \left(\ln |\{n : y_n = c\}| + \sum_{d=1}^{\mathsf{D}} \ln \frac{|\{n : x_{nd} = x_d, y_n = c\}|}{|\{n : y_n = c\}|} \right)
\end{aligned}$$

Naive Bayes

For **continuous features** with a Gaussian model,

$$\begin{aligned}
&\underset{c\in[\mathsf{C}]}{\operatorname{argmax}}\ p(y=c \mid \boldsymbol{x}) \\
&= \underset{c\in[\mathsf{C}]}{\operatorname{argmax}} \left(\ln p(y=c) + \sum_{d=1}^{\mathsf{D}} \ln p(x_d \mid y=c) \right) \\
&= \underset{c\in[\mathsf{C}]}{\operatorname{argmax}} \left(\ln |\{n : y_n = c\}| + \sum_{d=1}^{\mathsf{D}} \ln \left(\frac{1}{\sqrt{2\pi}\sigma_{cd}} \exp\left(-\frac{(x_d - \mu_{cd})^2}{2\sigma_{cd}^2} \right) \right) \right) \\
&= \underset{c\in[\mathsf{C}]}{\operatorname{argmax}} \left(\ln |\{n : y_n = c\}| - \sum_{d=1}^{\mathsf{D}} \left(\ln \sigma_{cd} + \frac{(x_d - \mu_{cd})^2}{2\sigma_{cd}^2} \right) \right)
\end{aligned}$$

Connection to logistic regression

Let us fix the variance for each feature to be σ (i.e. not a parameter of the model any more), then the prediction becomes

$$\begin{aligned}
&\underset{c\in[\mathsf{C}]}{\operatorname{argmax}}\ p(y=c \mid \boldsymbol{x}) \\
&= \underset{c\in[\mathsf{C}]}{\operatorname{argmax}} \left(\ln |\{n : y_n = c\}| - \sum_{d=1}^{\mathsf{D}} \left(\ln \sigma + \frac{(x_d - \mu_{cd})^2}{2\sigma^2} \right) \right) \\
&= \underset{c\in[\mathsf{C}]}{\operatorname{argmax}} \left(\ln |\{n : y_n = c\}| - \frac{\|\boldsymbol{x}\|_2^2}{2\sigma^2} - \sum_{d=1}^{\mathsf{D}} \frac{\mu_{cd}^2}{2\sigma^2} + \sum_{d=1}^{\mathsf{D}} \frac{\mu_{cd}}{\sigma^2} x_d \right) \\
&= \underset{c\in[\mathsf{C}]}{\operatorname{argmax}} \left(w_{c0} + \sum_{d=1}^{\mathsf{D}} w_{cd} x_d \right) = \underset{c\in[\mathsf{C}]}{\operatorname{argmax}}\ \boldsymbol{w}_c^{\mathrm{T}} \boldsymbol{x} \qquad (\textit{linear classifier!})
\end{aligned}$$

where we denote $w_{c0} = \ln |\{n : y_n = c\}| - \sum_{d=1}^{\mathsf{D}} \frac{\mu_{cd}^2}{2\sigma^2}$ and $w_{cd} = \frac{\mu_{cd}}{\sigma^2}$.

Connection to logistic regression

You can verify

$$p(y=c \mid x) \propto e^{\boldsymbol{w}_c^{\mathrm{T}} \boldsymbol{x}}$$

This is exactly the **softmax** function, the same model we used for a probabilistic interpretation of logistic regression!

So what is different then? They **learn the parameters in different ways**:

- both via MLE, one on $p(y=c \mid x)$, the other on $p(x, y)$
- solutions are different: logistic regression has no closed-form, naive Bayes admits a simple closed-form

Two different modeling paradigms

Suppose the training data is from an *unknown* joint probabilistic model $p(\boldsymbol{x}, y)$. There are two kinds of classification models in machine learning — generative models and discriminative models.

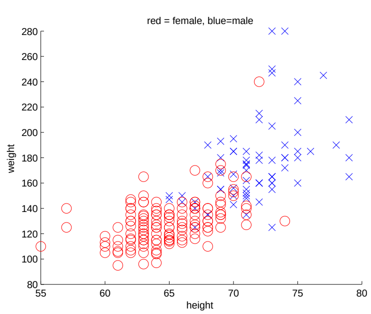
Differences in *assuming* models for the data

- the generative approach requires we specify the model for the joint distribution (such as Naive Bayes), and thus, maximize the *joint* likelihood $\sum_n \log p(\boldsymbol{x}_n, y_n)$
- the discriminative approach (discriminative) requires only specifying a model for the conditional distribution (such as logistic regression), and thus, maximize the *conditional* likelihood $\sum_n \log p(y_n|\boldsymbol{x}_n)$
- Sometimes, modeling by discriminative approach is easier
- Sometimes, parameter estimation by generative approach is easier

Generative model v.s discriminative model

	Discriminative model	Generative model
Example	logistic regression	naive Bayes
Model	conditional $p(y \mid x)$	joint $p(x, y)$ (might have same $p(y \mid x)$)
Learning	MLE	MLE
Accuracy	usually better for large N	usually better for small N
Remark		more flexible, can generate data after learning

Determining sex (man or woman) based on measurements

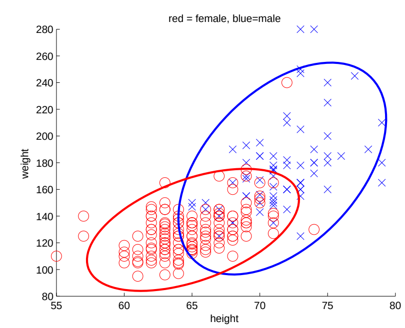


Example: Generative approach

Propose a model of the joint distribution of ($x =$ height, $y =$sex)

our data

Sex	Height
1	$6'$
2	$5'2"$
1	$5'6"$
1	$6'2"$
2	$5.7"$
$\cdots$	$\cdots$

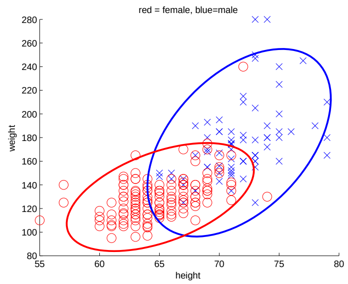


Intuition: we will model how heights vary (according to a Gaussian) in each sub-population (male and female).
Note: This is similar to Naive Bayes for detecting spam emails.

Model of the joint distribution

$$p(x, y) = p(y)p(x|y)$$

$$= \begin{cases} p_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} & \text{if } y = 1 \\ p_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}} & \text{if } y = 2 \end{cases}$$



where $p_1 + p_2 = 1$ represents two *prior* probabilities that x is given the label 1 or 2 respectively. $p(x|y)$ is assumed to be Gaussians.

Parameter estimation

Likelihood of the training data $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$ with $y_n \in \{1, 2\}$

$$\log P(\mathcal{D}) = \sum_n \log p(x_n, y_n)$$

$$= \sum_{n:y_n=1} \log \left(p_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x_n-\mu_1)^2}{2\sigma_1^2}} \right)$$

$$+ \sum_{n:y_n=2} \log \left(p_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x_n-\mu_2)^2}{2\sigma_2^2}} \right)$$

Maximize the likelihood function

$$(p_1^*, p_2^*, \mu_1^*, \mu_2^*, \sigma_1^*, \sigma_2^*) = \operatorname{argmax} \log P(\mathcal{D})$$

Decision boundary

The decision boundary between two classes is defined by

$$p(y = 1|x) \geq p(y = 2|x)$$

which is equivalent to

$$p(x|y = 1)p(y = 1) \geq p(x|y = 2)p(y = 2)$$

Namely,

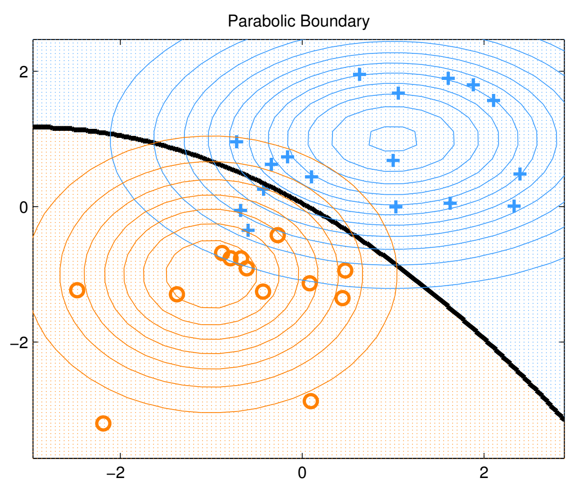
$$-\frac{(x - \mu_1)^2}{2\sigma_1^2} - \log \sqrt{2\pi}\sigma_1 + \log p_1 \geq -\frac{(x - \mu_2)^2}{2\sigma_2^2} - \log \sqrt{2\pi}\sigma_2 + \log p_2$$

It is quadratic in x. It follows (for some a, b and c, that

$$ax^2 + bx + c \geq 0$$

The decision boundary is *not linear*!

Example of nonlinear decision boundary



Note: the boundary is characterized by a quadratic function, giving rise to the shape of parabolic curve.

A special case

What if we assume the two Gaussians have the same variance?

We will get a *linear* decision boundary

From the previous slide:

$$-\frac{(x-\mu_1)^2}{2\sigma_1^2} - \log\sqrt{2\pi}\sigma_1 + \log p_1 \geq -\frac{(x-\mu_2)^2}{2\sigma_2^2} - \log\sqrt{2\pi}\sigma_2 + \log p_2$$

Setting $\sigma_1 = \sigma_2$, we obtain

$$bx + c \geq 0$$

Note: equal variances across two different categories could be a very strong assumption.

For example, the plot suggests that the *male* population has slightly bigger variance (i.e., bigger eclipse) than the *female* population.

Outline

Markov Models

Markov models are powerful **probabilistic models** to analyze sequential data. A.A.Markov (1856-1922) introduced the Markov chains in 1906 when he produced the first theoretical results for stochastic processes. They are now commonly used in

- text or speech recognition
- stock market prediction
- bioinformatics
- ...

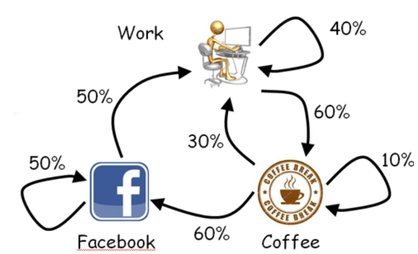
Markov chain

Directed strongly connected graph with self-loops.

Each edge labeled by a positive probability.

At each state, the probabilities on outgoing edges sum up to 1.

Transition (or stochastic) matrix: $A = a_{ij} = P(i \to j$ in 1 step).



$$A = \begin{array}{c|ccc} & W & C & F \\ \hline W & .4 & .6 & 0 \\ C & .3 & .1 & .6 \\ F & .5 & 0 & .5 \end{array}$$

Markov chain

Definition

Given a sequentially ordered random variables $X_1, X_2, \cdots, X_t, \cdots, X_T$, called **states**,

- **Transition probability** for describing how the state at time $t-1$ changes to the state at time t,

$$P(X_t = \text{value}' | X_{t-1} = \text{value})$$

- **Initial probability** for describing the initial state at time $t = 1$.

$$P(X_1 = \text{value})$$

All X_t's take value from the same discrete set $\{1, \ldots, N\}$.
We will assume that the transition probability does not change with respect to time t, i.e., a stationary Markov chain.

Markov chain

- Transition probabilities make a table/matrix $\boldsymbol{A}$ whose elements are

$$a_{ij} = P(X_t = j | X_{t-1} = i)$$

- Initial probability becomes a vector $\boldsymbol{\pi}$ whose elements are

$$\pi_i = P(X_1 = i)$$

where i or j index over from 1 to N. We have the following constraints

$$\sum_j a_{ij} = 1 \quad \sum_i \pi_i = 1$$

Additionally, all those numbers should be non-negative.

Examples

- Example 1 (**Language model**)

  States $[N]$ represent a dictionary of words,

$$a_{\text{ice,cream}} = P(X_{t+1} = \text{cream} \mid X_t = \text{ice})$$

  is an example of the transition probability.

- Example 2 (**Weather**)

  States $[N]$ represent weather at each day

$$a_{\text{sunny,rainy}} = P(X_{t+1} = \text{rainy} \mid X_t = \text{sunny})$$

Definition

A Markov chain is a stochastic process with the **Markov property**: a sequence of random variables $X_1, X_2, \ldots$ s.t.

$$P(X_{t+1} | X_1, X_2, \cdots, X_t) = P(X_{t+1} | X_t)$$

i.e. *the current state only depends on the most recent state*.

Is the Markov assumption reasonable? Not completely for the language model for example.

Higher order Markov chains make it more reasonable, e.g.

$$P(X_{t+1} | X_1, X_2, \cdots, X_t) = P(X_{t+1} \mid X_t, X_{t-1})$$

i.e. the current word only depends on the last two words.

Chain Rule

In all derivations we will be using the chain rule:

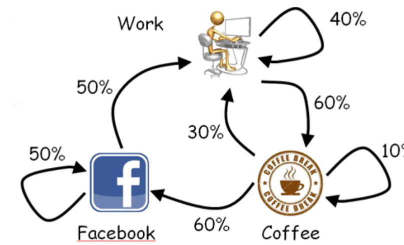
$$P(X, Y) = P(X \mid Y)\, P(Y) = P(Y \mid X)\, P(X)$$

$$P(X, Y, Z) = P(X, Y \mid Z)\, P(Z)$$

$$P(X, Y, Z) = P(X \mid Y, Z)\, P(Y \mid Z)\, P(Z)$$

Exercise 1

Consider the following Markov model. Given that now I am having Coffee, what's the probability that the next step is Facebook and the next is Work?



$P(X_3 = W, X_2 = F | X_1 = C) =$

$$= \frac{P(X_3 = W, X_2 = F, X_1 = C)}{P(X_1 = C)}$$

$$= \frac{P(X_3 = W | X_2 = F, X_1 = C) P(X_2 = F | X_1 = C) P(X_1 = C)}{P(X_1 = C)}$$

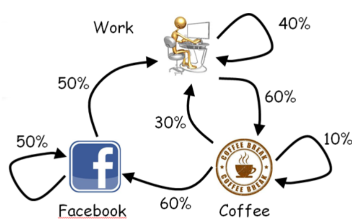
(chain rule)

$= P(X_3 = W | X_2 = F) P(X_2 = F | X_1 = C)$ (Markov rule)

$= 0.5 \times 0.6 = 0.3$

Exercise 2

Given that now I am having Coffee, what is the probability that in two steps I am at Work?



$P(X_3 = W | X_1 = C) =$

$= \sum_s P(X_3 = W, X_2 = s | X_1 = C) =$

$= P(X_3 = W | X_2 = W) P(X_2 = W | X_1 = C)$ (marginalization)

$+ P(X_3 = W | X_2 = C) P(X_2 = C | X_1 = C)$

$+ P(X_3 = W | X_2 = F) P(X_2 = F | X_1 = C)$

$= 0.3 \times 0.4 + 0.1 \times 0.3 + 0.6 \times 0.5 = 0.45$

Using a transition matrix:

$$P(X_3 = j | X_1 = i) = \sum_{k=1}^{N} a_{ik}\, a_{kj} = a^2_{ij}$$

Parameter estimation for Markov models

Now suppose we have observed M **sequences of examples**:

- $x_{1,1}, \ldots, x_{1,T}$
- $\cdots$
- $x_{M,1}, \ldots, x_{M,T}$

where

- for simplicity we assume each sequence has the same length T
- lower case $x_{n,t}$ represents the value of the random variable $X_{n,t}$

From these observations how do we *learn the model parameters* $(\boldsymbol{\pi}, \boldsymbol{A})$?

Finding the MLE

Same story, **Maximum Likelihood Estimation**:

$$\operatorname*{argmax}_{\boldsymbol{\pi}, \boldsymbol{A}} \ln P(X_1 = x_1, X_2 = x_2, \ldots, X_T = x_T)$$

First, we need to compute this joint probability. Applying the chain rule for random variables, we get

$$\begin{aligned}
&P(X_1, X_2, \ldots, X_T) \\
&= P(X_2, X_3, \ldots, X_T | X_1) P(X_1) \\
&= P(X_3, \ldots, X_T | X_1, X_2) P(X_2 | X_1) P(X_1) \\
&= \cdots = \\
&= P(X_1) \prod_{t=2}^{T} P(X_t | X_1, \ldots, X_{t-1}) \qquad \text{(Markov property)} \\
&= P(X_1) \prod_{t=2}^{T} P(X_t | X_{t-1})
\end{aligned}$$

Finding the MLE

The log-likelihood of a sequence $x_1, \ldots, x_T$ is

$$\begin{aligned}
&\ln P(X_1 = x_1, X_2 = x_2, \ldots, X_T = x_T) \\
&= P(X_1 = x_1) + \sum_{t=2}^{T} \ln P(X_t = x_t \mid X_{t-1} = x_{t-1}) \\
&= \ln \pi_{x_1} + \sum_{t=2}^{T} \ln a_{x_{t-1}, x_t} \\
&= \sum_{n} \mathbb{I}[x_1 = n] \ln \pi_n + \sum_{n,n'} \left(\sum_{t=2}^{T} \mathbb{I}[x_{t-1} = n, x_t = n'] \right) \ln a_{n,n'}
\end{aligned}$$

Finding the MLE

So MLE is

$$\operatorname*{argmax}_{\boldsymbol{\pi}, \boldsymbol{A}} \sum_{n} (\textbf{\#initial states with value } n) \ln \pi_n + \sum_{n,n'} (\textbf{\#transitions from } n \textbf{ to } n') \ln a_{n,n'}$$

We have seen this many times. The solution is (derivation is left as an exercise):

$$\pi_n = \frac{\#\text{of sequences starting with } n}{\#\text{of sequences}}$$

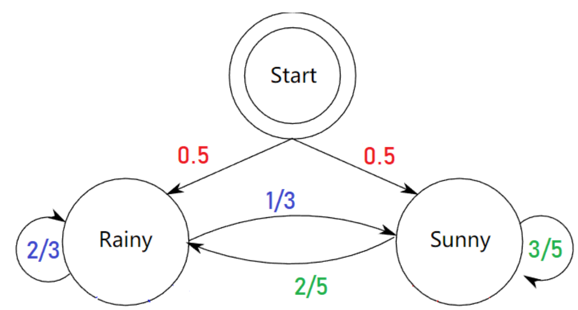
$$a_{n,n'} = \frac{\#\text{of transitions from } n \text{ to } n'}{\#\text{of transitions starting with } n}$$

Example

Suppose we observed the following 2 sequences of length 5

- sunny, sunny, rainy, rainy, rainy
- rainy, sunny, sunny, sunny, rainy

MLE is the following model



Outline

Markov Model with outcomes

Now suppose each state X_t also "emits" some **outcome** $O_t \in [O]$ based on the following model

$$P(O_t = o \mid X_t = s) = b_{s,o} \qquad \textbf{(emission probability)}$$

independent of anything else.

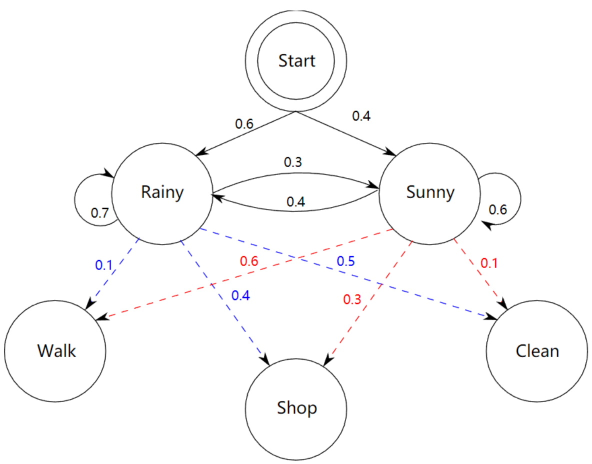
For example, in the language model, O_t is the speech signal for the underlying word X_t (very useful for speech recognition).

Now the model parameters are $(\{\pi_s\}, \{a_{s,s'}\}, \{b_{s,o}\}) = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$.

Another example

picture from Wikipedia

On each day, we also observe **Bob's activity: walk, shop, or clean**, which only depends on the weather of that day.



HMM defines a joint probability

$$\begin{aligned} &P(X_1, X_2, \cdots, X_T, O_1, O_2, \cdots, O_T) \\ &= P(X_1, X_2, \cdots, X_T)\, P(O_1, O_2, \cdots, O_T \mid X_1, X_2, \cdots, X_T) \end{aligned}$$

- Markov assumption simplifies the first term

$$P(X_1, X_2, \cdots, X_T) = P(X_1) \prod_{t=2}^{T} P(X_t \mid X_{t-1})$$

- The *independence* assumption simplifies the second term

$$P(O_1, O_2, \cdots, O_T \mid X_1, X_2, \cdots, X_T) = \prod_{t=1}^{T} P(O_t \mid X_t)$$

Namely, each O_t is conditionally independent of anything else, if conditioned on X_t.

Joint likelihood

The joint log-likelihood is

$$\ln P(X_1 = x_1, X_2 = x_2, \cdots, X_T = x_T, O_1 = o_1, O_2 = o_2, \cdots, O_T = o_T)$$

$$= \ln P(X_1 = x_1) \prod_{t=2}^{T} P(X_t = x_t \mid X_{t-1} = x_{t-1}) \prod_{t=1}^{T} P(O_t = o_t \mid X_t = x_t)$$

$$= \ln P(X_1 = x_1) + \sum_{t=2}^{T} \ln P(X_t = x_t \mid X_{t-1} = x_{t-1})$$

$$+ \sum_{t=1}^{T} \ln P(O_t = o_t \mid X_t = x_t)$$

$$= \ln \pi_{x_1} + \sum_{t=2}^{T} \ln a_{x_{t-1}, x_t} + \sum_{t=1}^{T} \ln b_{x_t, o_t}$$

Learning the model

If we observe M state-outcome sequences: $x_{m,1}, o_{m,1}, \ldots, x_{m,T}, o_{m,T}$ for $m = 1, \ldots, M$, the MLE is again very simple (verify yourself):

$$\pi_s \propto \textbf{\#initial states with value } s$$

$$a_{s,s'} \propto \textbf{\#transitions from } s \textbf{ to } s'$$

$$b_{s,o} \propto \textbf{\#state-outcome pairs } (s, o)$$

Learning the model

However, *most often we do not observe the states!* Think about the speech recognition example.

This is called **Hidden Markov Model (HMM)**.

Notice that "hidden" is referred to the states of the Markov chain, not to the parameters of the model.

A generic hidden Markov model is illustrated in this picture: