Course number: 80240743

Deep Learning

Xiaolin Hu (胡晓林) & Jun Zhu (朱军)
Dept. of Computer Science and Technology
Tsinghua University

A bit about the Instructor

- Jun Zhu, Professor, Depart. of Computer Science & Technology. I received my Ph.D. in DCST of Tsinghua University in 2009. My research interests include statistical machine learning, Bayesian nonparametrics, and data mining
- I did post-doc at the Machine Learning Department in CMU with Prof. Eric P. Xing. Before that I was invited to visit CMU for twice. I was also invited to visit Stanford for joint research (with Prof. Li Fei-Fei)
- 2015-2018: Adjunct Professor at CMU

- Published 100+ research papers on the top-tier ML conferences and journals, including JMLR, TPAMI, ICML, NIPS, etc.
- Served as Area Chairs for ICML, NIPS, UAI, AAAI, IJCAI; Associate Editor-in-Chief for PAMI
- Research is supported by National 973, NSFC, "Tsinghua 221 Basic Research Plan for Young Talents".
- Homepage: http://ml.cs.tsinghua.edu.cn/~jun

Schedule

No.	Date	Content	Instructor
10	May 12	Basics of generative models Homework 6	Jun Zhu
11	May 19	Variational Auto-Encoders	Jun Zhu
12	May 26	Generative adversarial networks Homework 7	Jun Zhu
13	June 2	Generative flows and ZhuSuan library	Jun Zhu
15	June 6	Project presentation	Xiaolin Hu & Jun Zhu

Basics of Generative Models

Jun Zhu

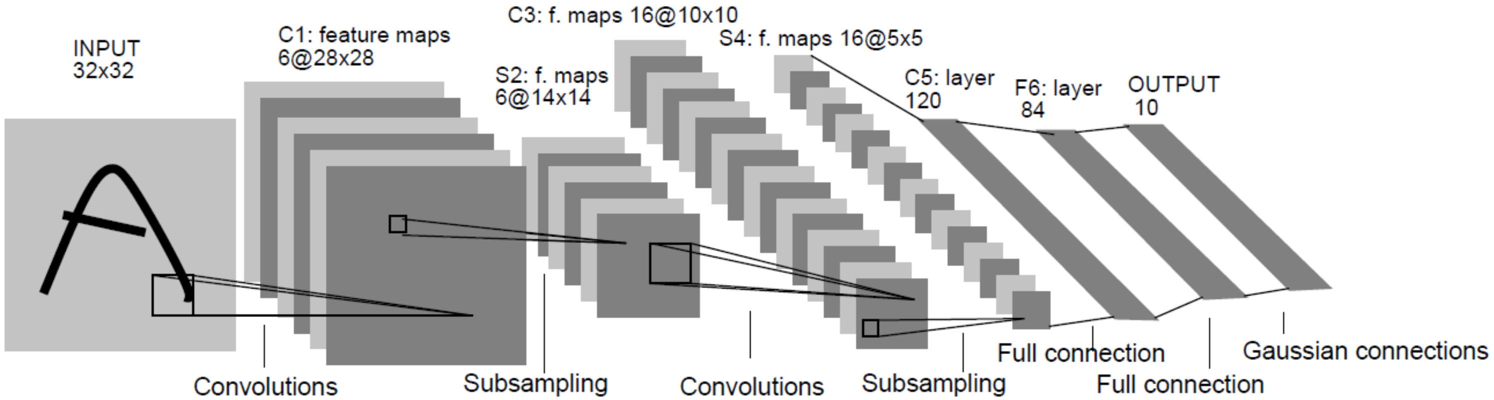
dcszj@mail.tsinghua.edu.cn

Department of Computer Science and Technology

Tsinghua University

Discriminative Deep Learning

- Learn a deep NN to map an input to output



- Gradient back-propagation
- Dropout
- Activation functions:
  - rectified linear

Generative Modeling

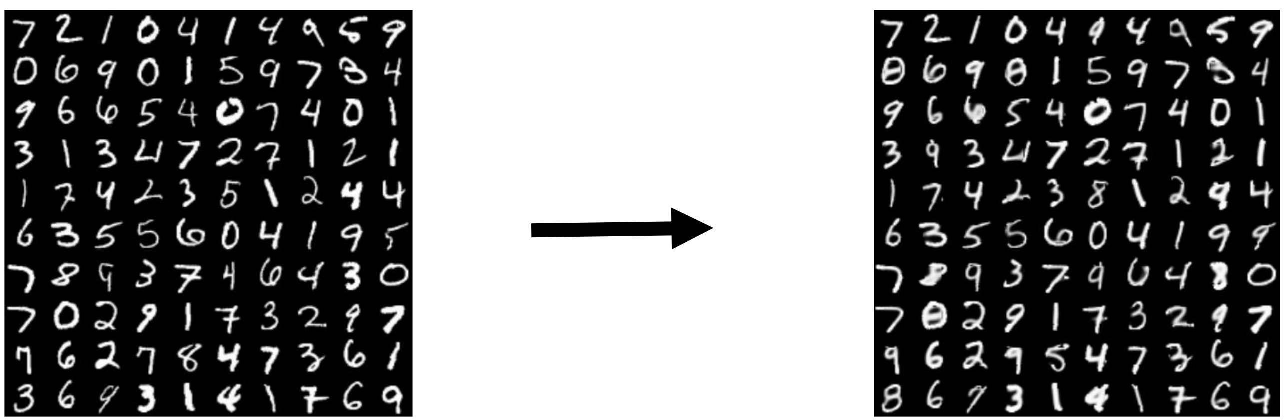
- Have training examples

$$x \sim p_{\text{data}}(x)$$

- Want a model that can draw samples:

$$x' \sim p_{\text{model}}(x)$$

  - where $p_{\text{model}}(x) \approx p_{\text{data}}(x)$



$x \sim p_{\text{data}}(x)$ $\qquad\qquad$ $x' \sim p_{\text{model}}(x)$

Why generative models?

“What I cannot create, I do not understand.”

—Richard Feynman

Why generative models?

- Leverage unlabeled datasets, which are often much larger than labeled ones
  - Unsupervised learning
  - Semi-supervised learning

- Conditional generative models
  - Speech synthesis: Text $\Rightarrow$ Speech
  - Machine Translation: French $\Rightarrow$ English
  - Image captioning: Image $\Rightarrow$ Text

Outline

- Review of Probability and Statistics
  - MLE
- Generative Models
- EM algorithms

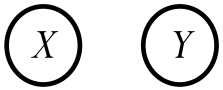
Basics of Probabilities and MLE

Independence

- Independent random variables:

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

$$P(X|Y) = P(X)$$



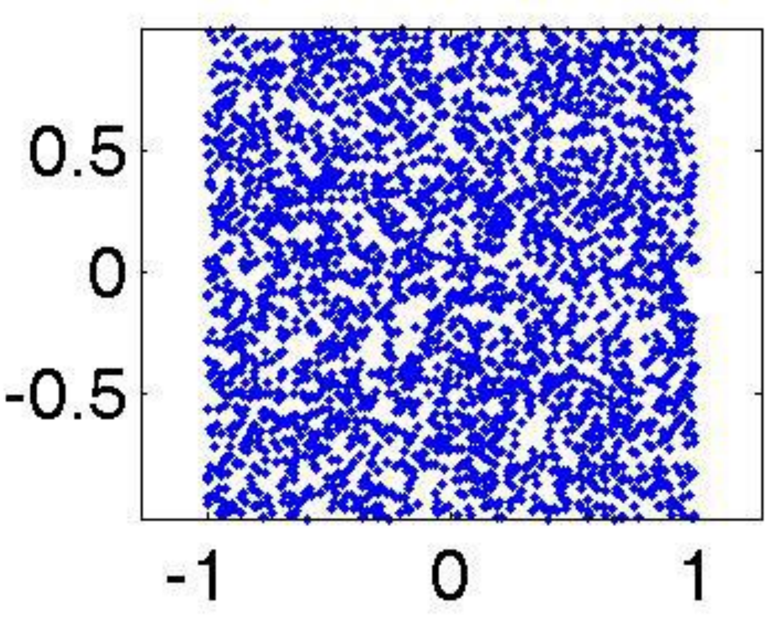
  - Y and X don't contain information about each other

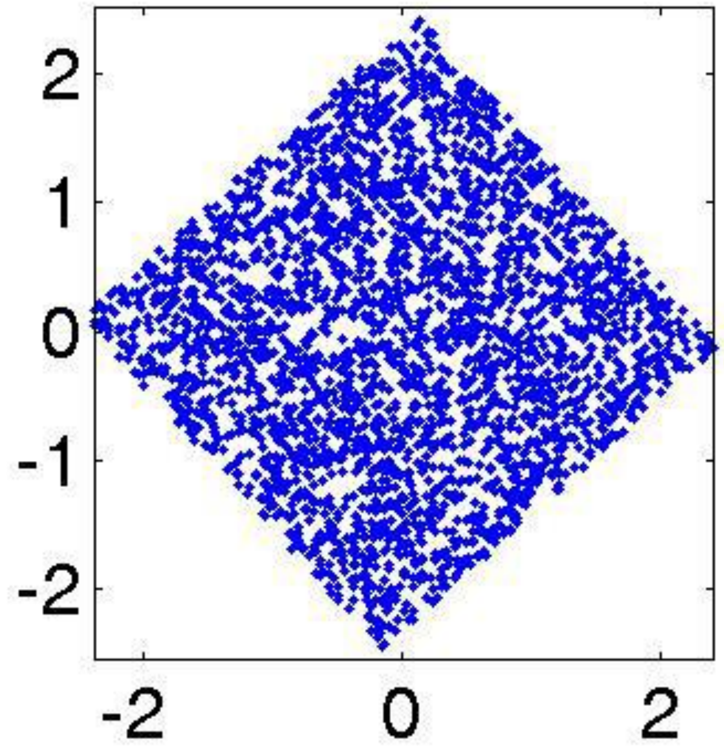
    Observing Y doesn't help predicting X

    Observing X doesn't help predicting Y

- Examples:
  - Independent:
    - winning on roulette this week and next week
  - Dependent:
    - Russian roulette

Dependent / Independent?

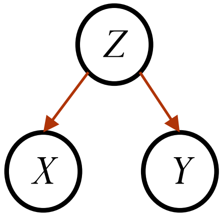




Conditional Independence

- Conditionally independent:

$$P(X,Y|Z) = P(X|Z)P(Y|Z)$$



  - knowing Z makes X and Y independent

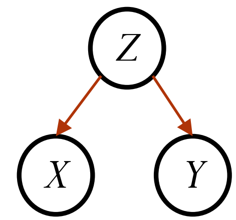
- Examples:

London taxi drivers: A survey has pointed out a positive and significant correlation between the number of accidents and wearing coats. They concluded that coats could hinder movements of drivers and be the cause of accidents. A new law was prepared to prohibit drivers from wearing coats when driving.

Finally another study pointed out that people wear coats when it rains…

Conditional Independence

- Conditionally independent:



$$P(X,Y|Z) = P(X|Z)P(Y|Z)$$

  - knowing Z makes X and Y independent

- Equivalent to:

$$\forall(x,y,z)\colon\ P(X=x|Y=y,Z=z) = P(X=x\,|Z=z)$$

  - E.g.:

$$P(Thunder\mid Rain, Lighting) = \mathrm{P}(Thunder\mid Lighting)$$

Maximum Likelihood Estimation (MLE)

Flipping a Coin

- What's the probability that a coin will fall with a head up (if flipped)?
- Let us flip it a few times to estimate the probability

The estimated probability is: 3/5 "frequency of heads"

Questions:

The estimated probability is: 3/5 "frequency of heads"

- Why frequency of heads?
- How good is this estimation?

Question (1)

- Why frequency of heads?
  - Frequency of heads is exactly the Maximum Likelihood Estimator for this problem

  - MLE has nice properties

    (interpretation, statistical guarantees, simple)

MLE for Bernoulli Distribution

Data, D =

$$D = \{X_i\}_{i=1}^{n}, \ X_i \in \{\mathrm{H}, \mathrm{T}\}$$

$$P(Head) = \theta \qquad P(Tail) = 1 - \theta$$

- Flips are i.i.d:
  - **Independent** events that are **identically distributed** according to Bernoulli distribution
- **MLE**: choose θ that maximizes the probability of observed data

Maximum Likelihood Estimation (MLE)

- MLE: choose θ that maximizes the probability of observed data

$$\hat{\theta}_{MLE} = \arg\max_{\theta} P(D|\theta)$$

$$= \arg\max_{\theta} \prod_{i=1}^{n} P(X_i|\theta)$$ Independent draws

$$= \arg\max_{\theta} \prod_{i:X_i=H} \theta \prod_{i:X_i=T} (1-\theta)$$ Identically distributed

$$= \arg\max_{\theta} \theta^{N_H} (1-\theta)^{N_T}$$

Maximum Likelihood Estimation (MLE)

- MLE: choose θ that maximizes the probability of observed data

$$\hat{\theta}_{MLE} = \arg\max_{\theta} P(D|\theta)$$

$$= \arg\max_{\theta} \theta^{N_H}(1-\theta)^{N_T}$$

- Solution?

$$\hat{\theta}_{MLE} = \frac{N_H}{N_H + N_T}$$

  - Exactly the "**Frequency of heads**"

Question (2)

- How good is the MLE estimation?

$$\hat{\theta}_{MLE} = \frac{N_H}{N_H + N_T}$$

  - Is it biased?

How many flips do I need?

- I flipped the coins 5 times: 3 heads, 2 tails

$$\hat{\theta}_{MLE} = \frac{3}{5}$$

- What if I flipped 30 heads and 20 tails?

$$\hat{\theta}_{MLE} = \frac{30}{50}$$

- Which estimator should we trust more?

A Simple Bound

- Let $\theta^\star$ be the true parameter. For n data points, and

$$\hat{\theta}_{MLE} = \frac{N_H}{N_H + N_T}$$

- Then, for any ε>0 , we have the Hoeffding's Inequality:

$$P(|\hat{\theta} - \theta^\star| \geq \epsilon) \leq 2e^{-2n\epsilon^2}$$

Probably Approximately Correct (PAC) Learning

- I want to know the coin parameter θ, within ε=0.1 error with probability at least 1-δ (e.g., 0.95)
- How many flips do I need?

$$P(|\hat{\theta} - \theta^{\star}| \geq \epsilon) \leq 2e^{-2n\epsilon^2} \leq \delta$$

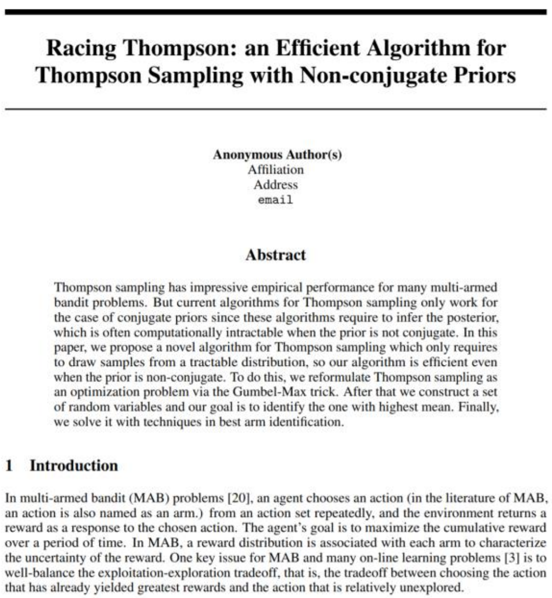
- Sample complexity:

$$n \geq \frac{\ln(2/\delta)}{2\epsilon^2}$$

Examples – Language Model

- A simple unigram language model
  - Observations (e.g., bag-of-words)

$$\mathrm{x} = \{x_1, \ldots, x_d\}$$



Racing Thompson: an Efficient Algorithm for Thompson Sampling with Non-conjugate Priors

Anonymous Author(s)
Affiliation
Address
email

Abstract

Thompson sampling has impressive empirical performance for many multi-armed bandit problems. But current algorithms for Thompson sampling only work for the case of conjugate priors since these algorithms require to infer the posterior, which is often computationally intractable when the prior is not conjugate. In this paper, we propose a novel algorithm for Thompson sampling which only requires to draw samples from a tractable distribution, so our algorithm is efficient even when the prior is non-conjugate. To do this, we reformulate Thompson sampling as an optimization problem via the Gumbel-Max trick. After that we construct a set of random variables and our goal is to identify the one with highest mean. Finally, we solve it with techniques in best arm identification.

1 Introduction

In multi-armed bandit (MAB) problems [20], an agent chooses an action (in the literature of MAB, an action is also named as an arm.) from an action set repeatedly, and the environment returns a reward as a response to the chosen action. The agent's goal is to maximize the cumulative reward over a period of time. In MAB, a reward distribution is associated with each arm to characterize the uncertainty of the reward. One key issue for MAB and many on-line learning problems [3] is to well-balance the exploitation-exploration tradeoff, that is, the tradeoff between choosing the action that has already yielded greatest rewards and the action that is relatively unexplored.

Term	D1	D2
game	1	0
decision	0	0
theory	2	0
probability	0	3
analysis	0	2
…		

Examples – Language Model

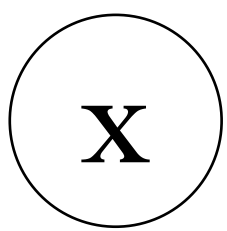
- A simple unigram language model
  - Observations (e.g., bag-of-words)

$$\mathrm{x} = \{x_1, \ldots, x_d\}$$

  - Joint distribution (likelihood)

$$p(\mathrm{x}; \theta) = \prod p(x_i \, ; \theta)$$

  - Graphical representation (parameters ignored)



Examples – Language Model

- Learn a simple generative model
  - Given a set of observations

$$\mathrm{X} = \{\mathrm{x}_1, \ldots, \mathrm{x}_N\}$$

  - Maximize the log-likelihood

$$\max_{\theta} \log p(\mathrm{X}; \theta) = \sum \log p(\mathrm{x}_\mathrm{i}\,; \theta)$$

  - Simple closed-form solutions:
    - count frequency for discrete or empirical mean/variance for Gaussian distribution

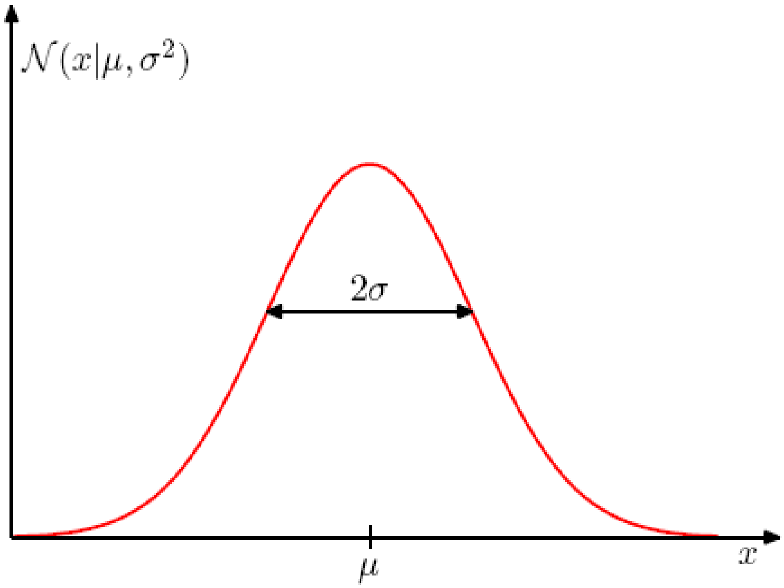
Examples – Gaussian Model

Carl F. Gauss (1777 – 1855)

- Univariate Gaussian distribution

$$p(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

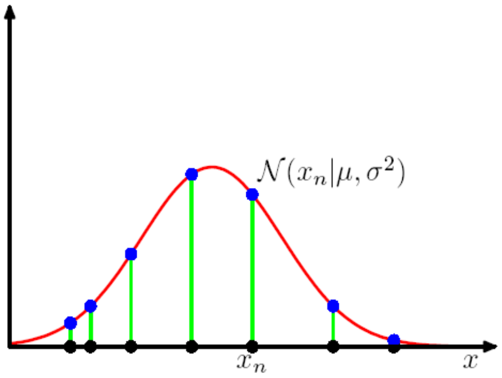
- Given parameters, we can draw samples and plot distributions



Maximum Likelihood Estimation

- Given a data set $\mathcal{D} = \{x_1, \ldots, x_N\}$, the likelihood is

$$p(\mathcal{D}|\mu, \sigma^2) = \prod_{n=1}^{N} p(x_n|\mu, \sigma^2)$$



- MLE estimates the parameters as

$$(\mu_{\mathrm{ML}}, \sigma^2_{\mathrm{ML}}) = \operatorname*{argmax}_{\mu, \sigma^2} \log p(\mathcal{D}|\mu, \sigma^2)$$

$$\mu_{\mathrm{ML}} = \frac{1}{N} \sum_{n=1}^{N} x_n$$ sample mean

$$\sigma^2_{\mathrm{ML}} = \frac{1}{N} \sum_{n=1}^{N} (x_n - \mu_{\mathrm{ML}})^2$$ sample variance

Note: MLE for the variance of a Gaussian is biased

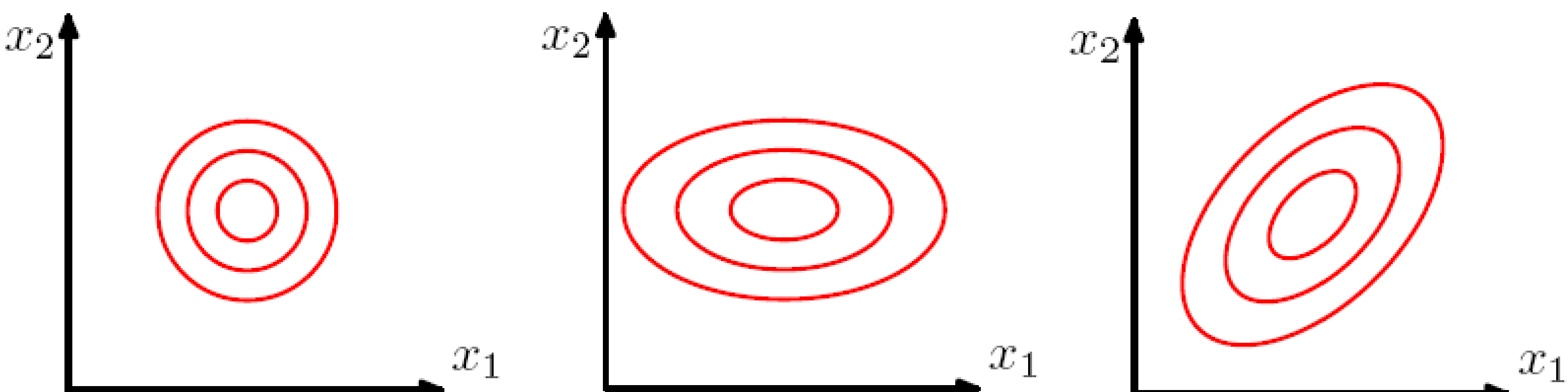
Gaussian Distributions

Carl F. Gauss (1777 – 1855)

- d-dimensional multivariate Gaussian

$$p(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(- \frac{1}{2}(\mathbf{x} - \mu)\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}) \right)$$

- Given parameters, we can draw samples and plot distributions



Isotropic

$$\Sigma = \sigma^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Diagonal

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$

General

$$\Sigma = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

Maximum Likelihood Estimation

- Given a data set $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, the likelihood is

$$p(\mathcal{D}|\mu, \Sigma) = \prod_{n=1}^{N} p(\mathbf{x}_n|\mu, \Sigma)$$

- MLE estimates the parameters as

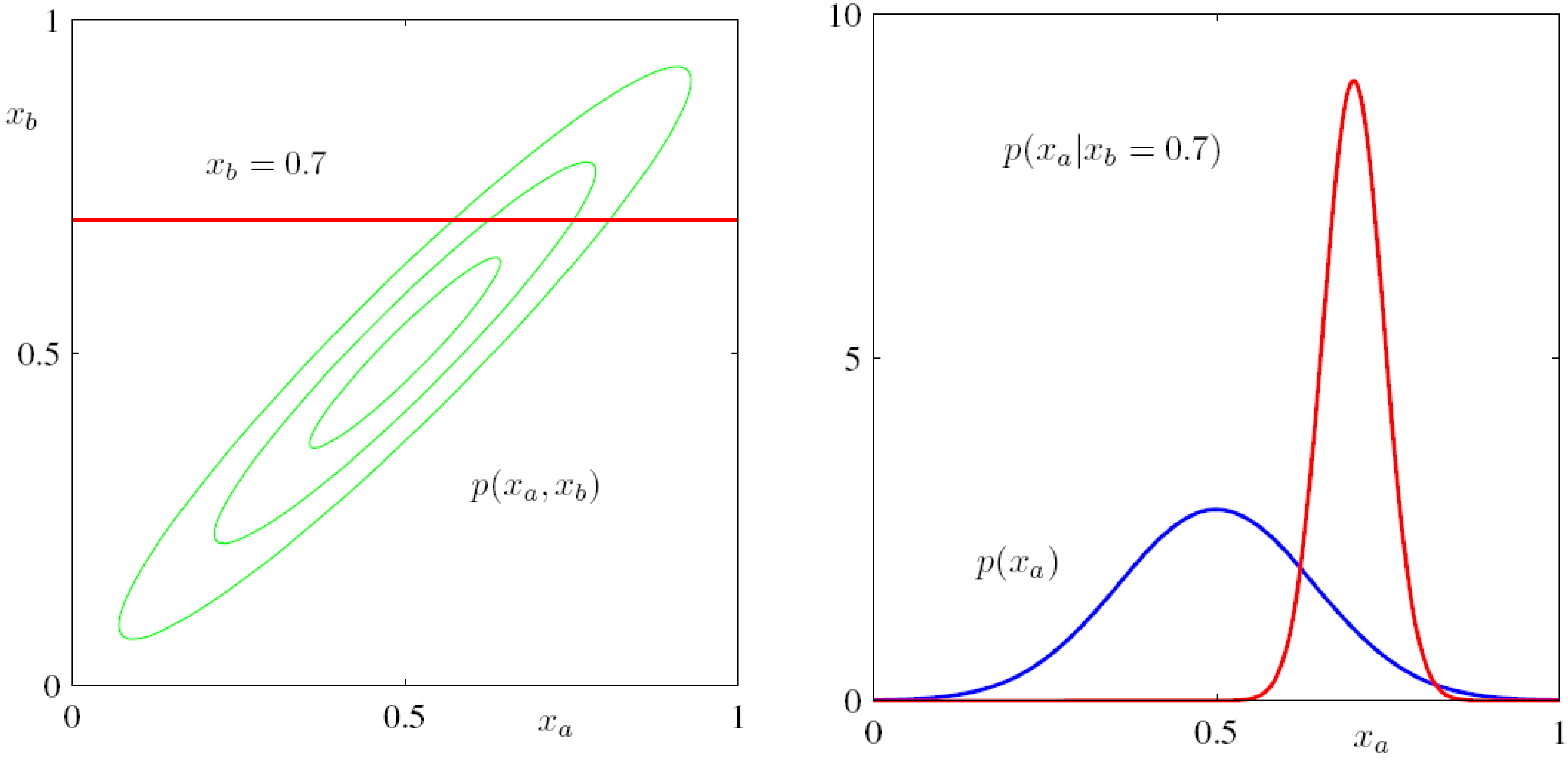
$$(\mu_{\mathrm{ML}}, \Sigma_{\mathrm{ML}}) = \operatorname*{argmax}_{\mu, \Sigma} \log p(\mathcal{D}|\mu, \Sigma)$$

$$\mu_{\mathrm{ML}} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n$$ sample mean

$$\Sigma_{\mathrm{ML}}^2 = \frac{1}{N} \sum_{n=1}^{N} (x_n - \mu_{\mathrm{ML}})(x_n - \mu_{\mathrm{ML}})^\top$$ sample covariance

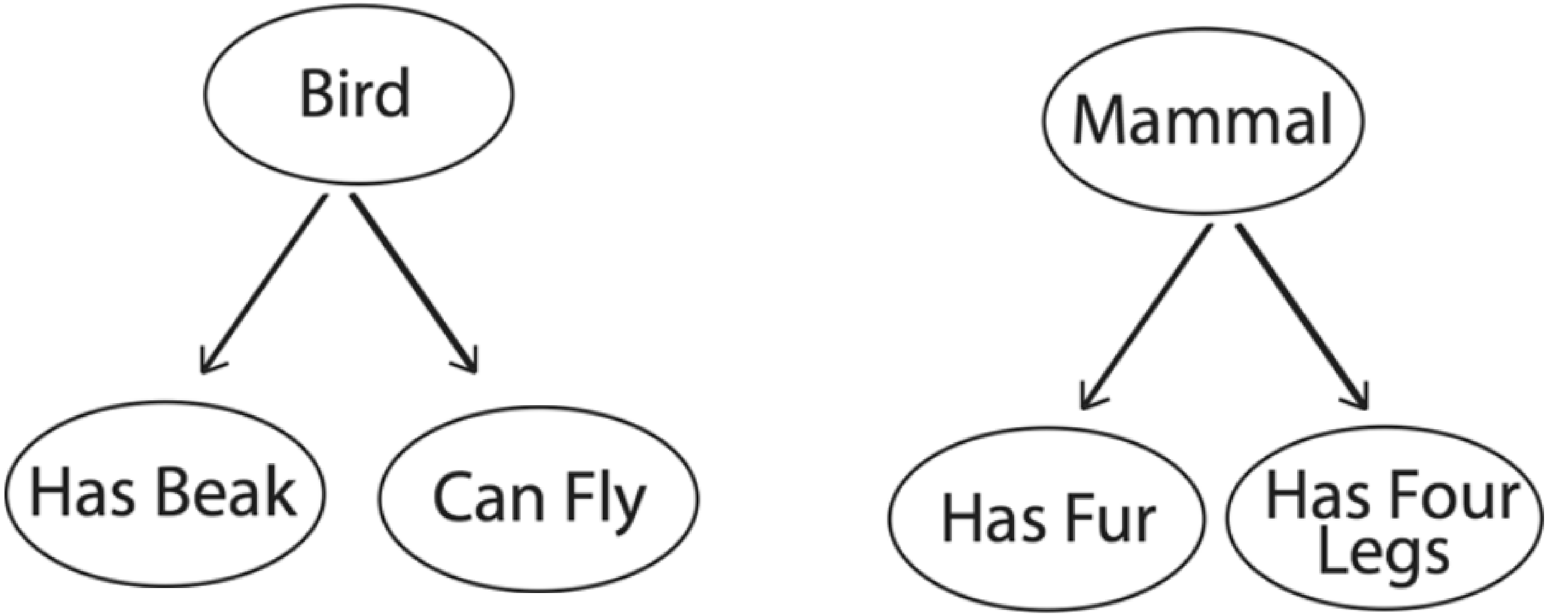
Other Nice Analytic Properties

- Marginal is Gaussian
- Conditional is Gaussian



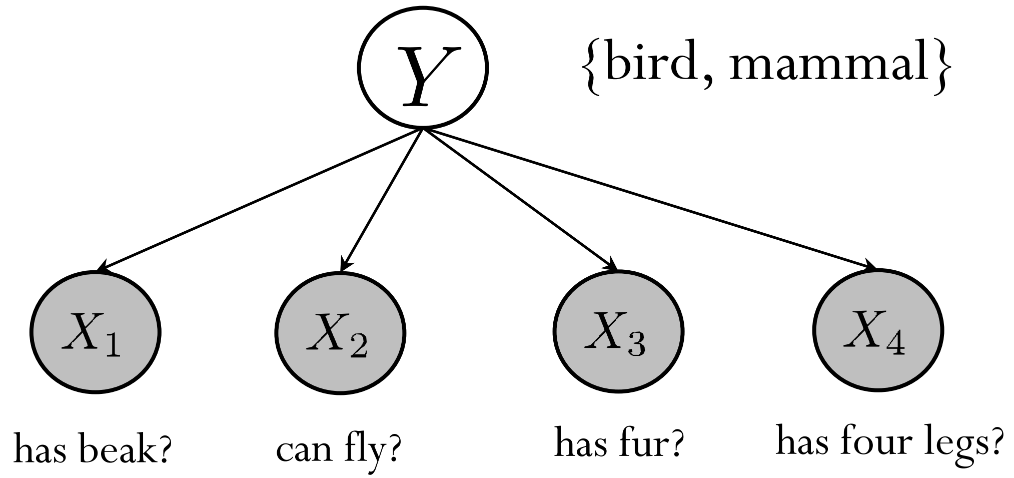
Example – Naïve Bayes Classifier

- The simplest "**category-feature**" generative model:
  - **Category**: "bird", "Mammal"
  - **Features**: "has beak", "can fly" …



Naïve Bayes Classifier

- **A mathematatic model**:
  - **Naive Bayes assumption**: features $X_1, \ldots, X_d$ are conditionally independent given the class label Y



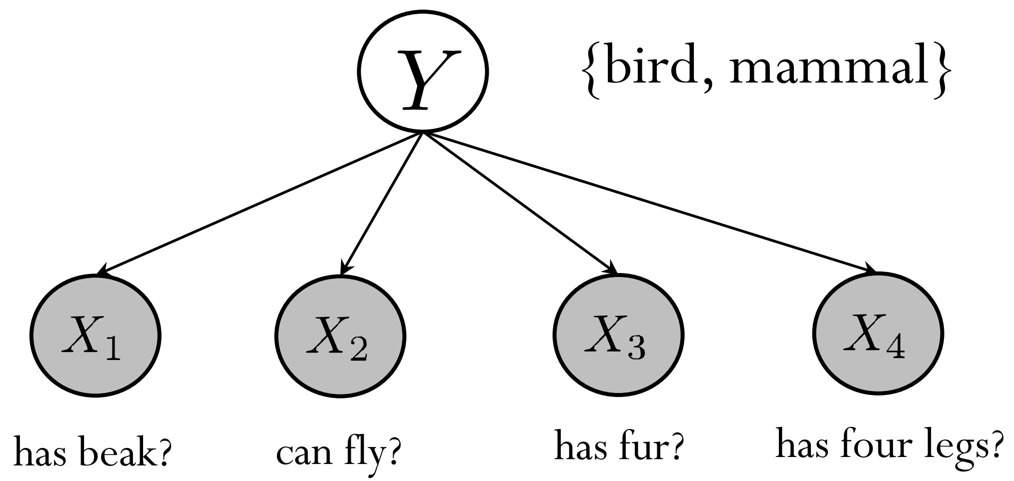
A joint distribution:

prior likelihood

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

Naïve Bayes Classifier

- **A mathematic model:**



Inference via Bayes rule:

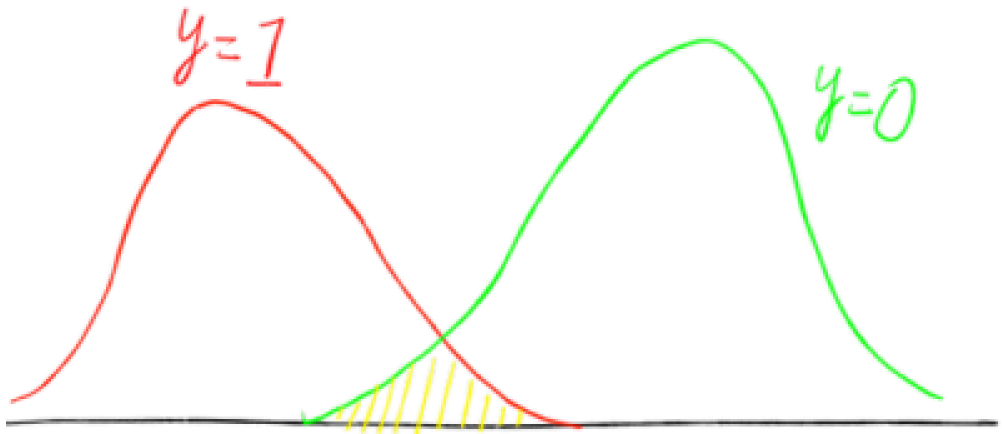
$$p(y|\mathrm{x}) = \frac{p(\mathrm{x}, y)}{p(\mathrm{x})} = \frac{p(y)p(\mathrm{x}|y)}{p(\mathrm{x})}$$

Bayes' decision rule:

$$y^* = \arg\max_{y \in \mathcal{Y}} p(y|\mathrm{x})$$

Bayes Error

- **Theorem**: Bayes classifier is optimal!



$$p(error|\mathbf{x}) = \begin{cases} p(y=1|\mathbf{x}) & \text{if we decide } y=0 \\ p(y=0|\mathbf{x}) & \text{if we decide } y=1 \end{cases}$$

$$p(error) = \int_{-\infty}^{\infty} p(error|\mathbf{x})p(\mathbf{x})d\mathbf{x}$$

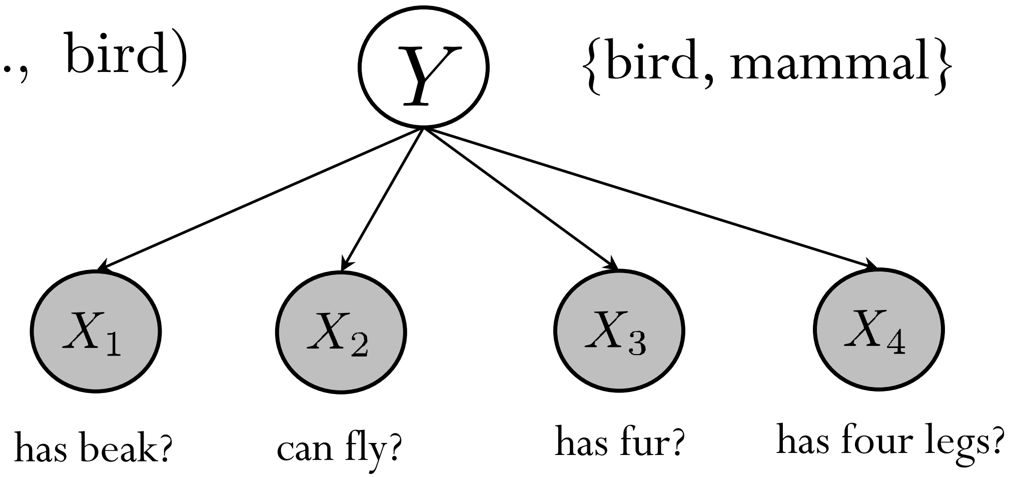
- *However, the true distribution is unknown.*

- *Learning!*
  - *We need to estimate it!*

Naïve Bayes Classifier

- **How to learn model parameters?**
  - Assume X are d binary features, Y has 2 possible labels

$$p(y|\pi) = \begin{cases} \pi & \text{if } y = 1 \ (i.e., \ \text{bird}) \\ 1 - \pi & \text{otherwise} \end{cases}$$



$$p(x_j|y = 0, q) = \begin{cases} q_{0j} & \text{if } x_j = 1 \\ 1 - q_{0j} & \text{otherwise} \end{cases} \qquad p(x_j|y = 1, q) = \begin{cases} q_{1j} & \text{if } x_j = 1 \\ 1 - q_{1j} & \text{otherwise} \end{cases}$$

  - *How many parameters to estimate?*

Naïve Bayes Classifier

- **How to learn model parameters?**
- **A set of training data:**
  - (1, 1, 0, 0; 1)
  - (1, 0, 0, 0; 1)
  - (0, 1, 1, 0; 0)
  - (0, 0, 1, 1; 0)
- **Maximum likelihood estimation** (N: # of training data)

$$p(\{\mathbf{x}_i, y_i | \pi, q\}) = \prod_{i=1}^{N} p(\mathbf{x}_i, y_i | \pi, q)$$

Naïve Bayes Classifier

- **Maximum likelihood estimation** (N: # of training data)

$$(\hat{\pi}, \hat{q}) = \arg\max_{\pi, q} p(\{\mathbf{x}_i, y_i\} | \pi, q)$$

$$(\hat{\pi}, \hat{q}) = \arg\max_{\pi, q} \log p(\{\mathbf{x}_i, y_i\} | \pi, q)$$

- **Results (**count frequency! Exercise?**)**:

$$\hat{\pi} = \frac{N_1}{N} \qquad \hat{q}_{0j} = \frac{N_0^j}{N_0} \qquad \hat{q}_{1j} = \frac{N_1^j}{N_1}$$

$$N_k = \sum_{i=1}^{N} \mathbf{I}(y_i = k) : \ \text{\# of data in category } k$$

$$N_k^j = \sum_{i=1}^{N} \mathbf{I}(y_i = k, \ x_{ij} = 1) : \ \text{\# of data in category } k \text{ that has feature } j$$

Naïve Bayes Classifier

- Data scarcity issue (zero-counts problem):

$$\hat{\pi} = \frac{N_1}{N} \qquad \hat{q}_{0j} = \frac{N_0^j}{N_0} \qquad \hat{q}_{1j} = \frac{N_1^j}{N_1}$$

  - ***How about if some features do not appear?***

- Laplace smoothing (Additive smoothing):

$$\hat{q}_{0j} = \frac{N_0^j + \alpha}{N_0 + 2\alpha}$$

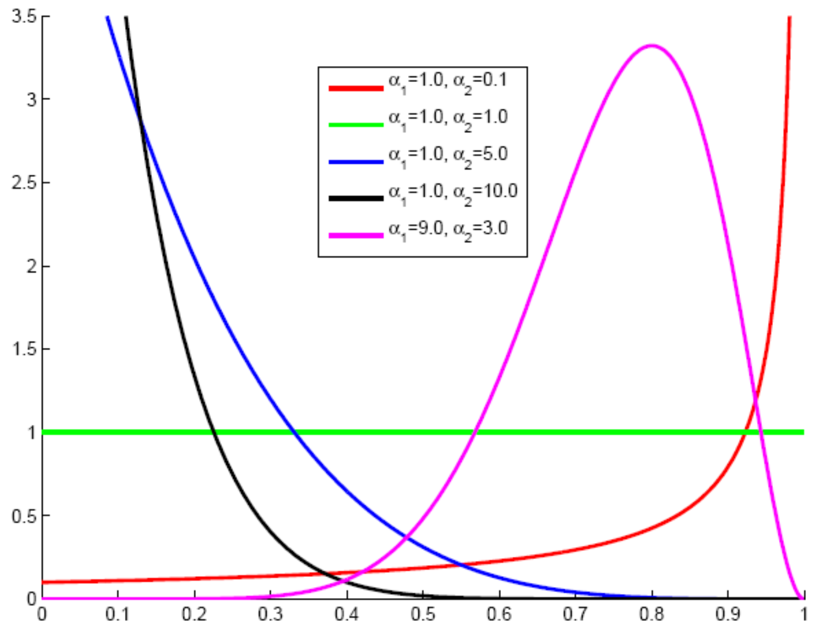
$$\alpha > 0$$

$$\hat{q}_{1j} = \frac{N_1^j + \alpha}{N_1 + 2\alpha}$$

A Bayesian Treatment

- Put a prior on the parameters

$$p_0(q_{0j}|\alpha_1,\alpha_2) = \mathrm{Beta}(\alpha_1,\alpha_2) = \frac{\Gamma(\alpha_1+\alpha_2)}{\Gamma(\alpha_1)\Gamma(\alpha_2)} q_{0j}^{\alpha_1-1}(1-q_{0j})^{\alpha_2-1}$$



A Bayesian Treatment

- Maximum a Posterior Estimate (MAP):

$$\hat{q} = \arg\max_q \log p(q|\{\mathbf{x}_i, y_i\})$$
$$= \arg\max_q \log p_0(q) + \log p(\{\mathbf{x}_i, y_i\}|q)$$

- Results (Exercise?):

$$\hat{q}_{0j} = \frac{N_0^j + \alpha_1 - 1}{N_0 + \alpha_1 + \alpha_2 - 2}$$

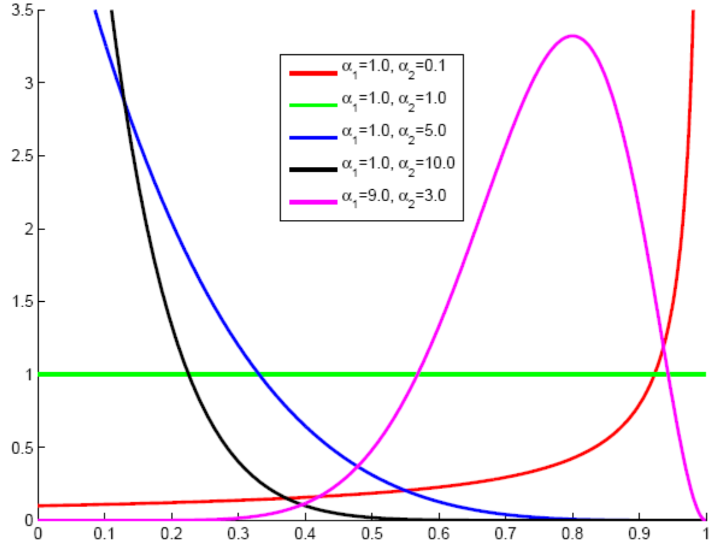
$$\hat{q}_{1j} = \frac{N_1^j + \alpha_1 - 1}{N_1 + \alpha_1 + \alpha_2 - 2}$$

A Bayesian Treatment

- Maximum a Posterior Estimate (MAP):

$$\hat{q}_{0j} = \frac{N_0^j + \alpha_1 - 1}{N_0 + \alpha_1 + \alpha_2 - 2}$$

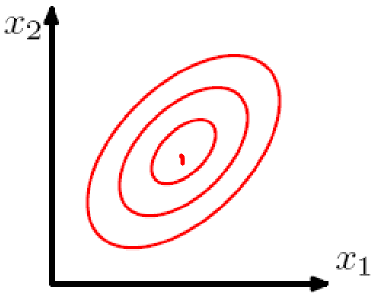
- If $\alpha_1 = \alpha_2 = 1$ (non-informative prior), no effect
  - MLE is a special case of Bayesian estimate
- Increase α_1, α_2 , lead to heavier influence from prior



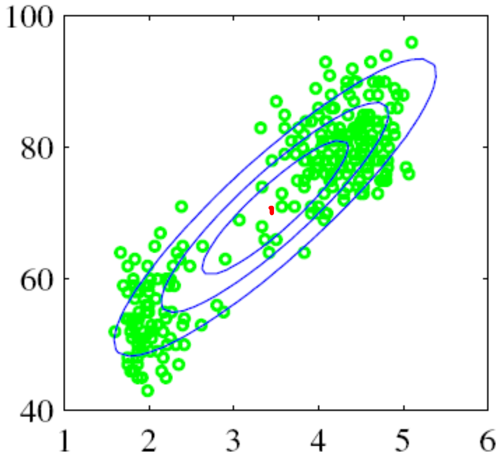
Generative Models with Latent Variables and EM Algorithms

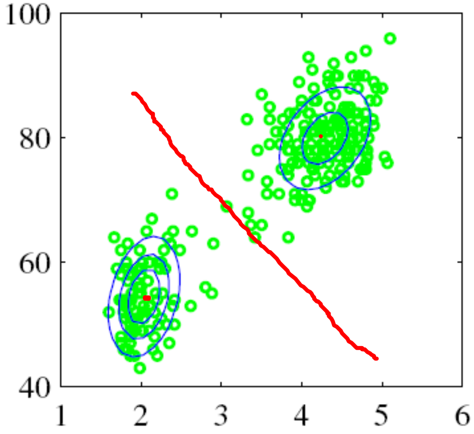
Limitations of Single Gaussians

- Single Gaussian is unimodal



- … can't fit well multimodal data, which is more realistic!

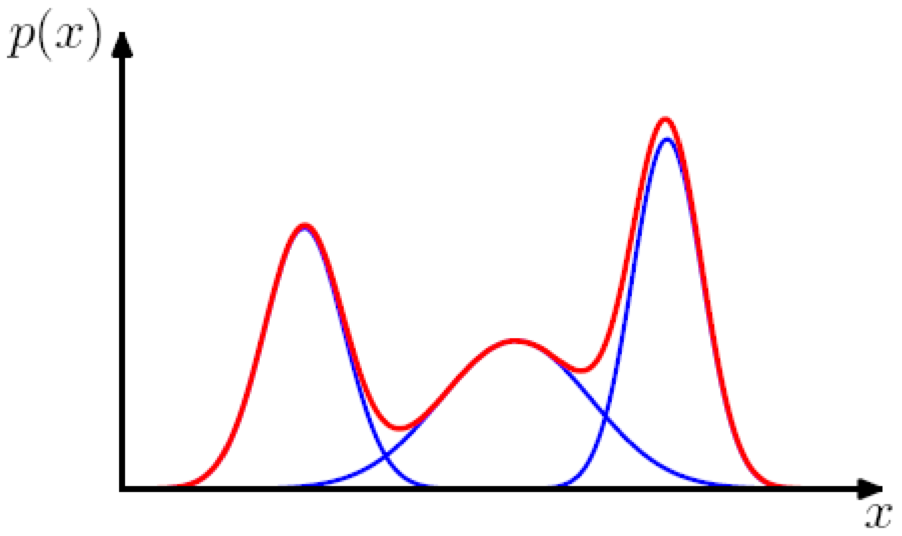




Mixture of Gaussians

- A simple family of multi-modal distributions
  - treat unimodal Gaussians as basis (or component) distributions
  - superpose multiple Gaussians via linear combination

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \sigma_k^2)$$

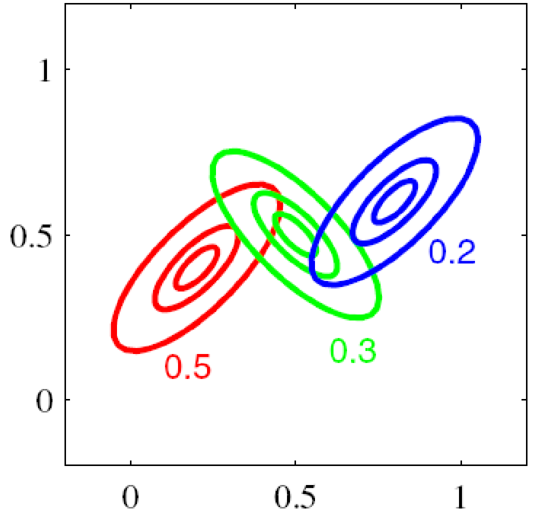


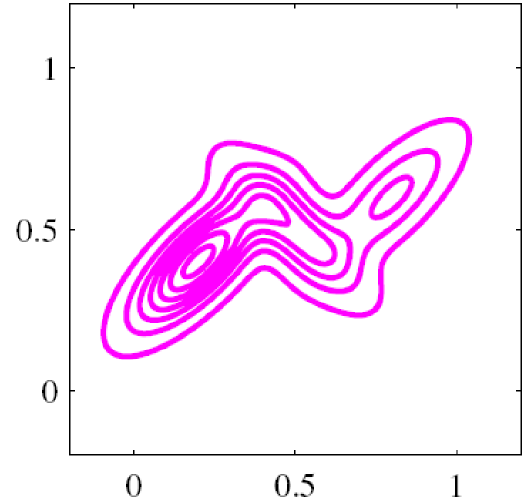
Mixture of Gaussians

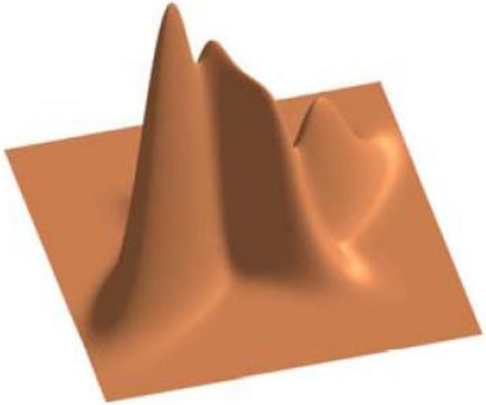
- A simple family of multi-modal distributions
  - treat unimodal Gaussians as basis (or component) distributions
  - superpose multiple Gaussians via linear combination

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)$$

What conditions should the mixing coefficients satisfy ?







MLE for Mixture of Gaussians

- Log-likelihood

$$\log p(\mathcal{D}|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k) \right)$$

  - this is complicated … ☹
  - … but, we know the MLE for single Gaussians are easy

- A heuristic procedure (can we iterate?)
  - allocate data into different components
  - estimate each component Gaussian analytically

Optimal Conditions

- Some math

$$\mathcal{L}(\boldsymbol{\mu}, \Sigma) = \log p(\mathcal{D}|\boldsymbol{\mu}, \Sigma) = \sum_{n=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k) \right)$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} = 0 \quad \Rightarrow \quad \sum_{n=1}^{N} \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \Sigma_j)}}_{\gamma(z_{nk})} \Sigma_k^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_k) = 0$$

$$\Rightarrow \quad \boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \qquad N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

A weighted sample mean!

Optimal Conditions

- Some math

$$\mathcal{L}(\boldsymbol{\mu}, \Sigma) = \log p(\mathcal{D}|\boldsymbol{\mu}, \Sigma) = \sum_{n=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k) \right)$$

$$\frac{\partial \mathcal{L}}{\partial \Sigma_k} = 0 \quad \Rightarrow \quad \Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top$$

A weighted sample variance!

Optimal Conditions

- Some math

$$\mathcal{L}(\boldsymbol{\mu}, \Sigma) = \log p(\mathcal{D}|\boldsymbol{\mu}, \Sigma) = \sum_{n=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k) \right)$$

Note: constraints exist for mixing coefficients!

$$L = \mathcal{L}(\boldsymbol{\mu}, \Sigma) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right)$$

$$\frac{\partial L}{\partial \pi_k} = 0 \quad \Rightarrow \quad \sum_{n=1}^{N} \frac{\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \Sigma_j)} + \lambda = 0$$

$$\Rightarrow \quad \pi_k = \frac{N_k}{N}$$

The ratio of data assigned to component k!

Optimal Conditions – summary

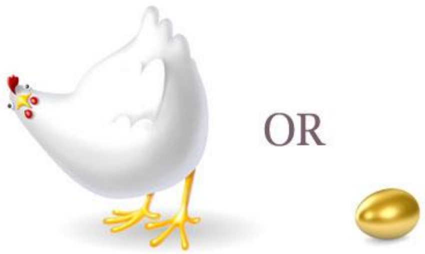
- The set of couple conditions

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top$$

$$\pi_k = \frac{N_k}{N}$$

- The key factor to get them coupled



$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \Sigma_j)}$$

- If we know $\gamma(z_{nk})$, each component Gaussian is easy to estimate!

The EM Algorithm

- **E-step**: estimate the responsibilities

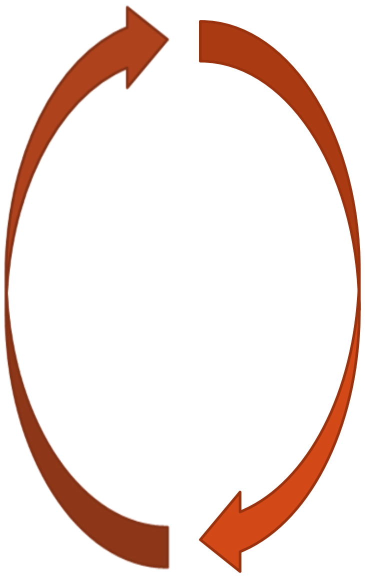
$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \Sigma_j)}$$

- **M-step**: re-estimate the parameters

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$$

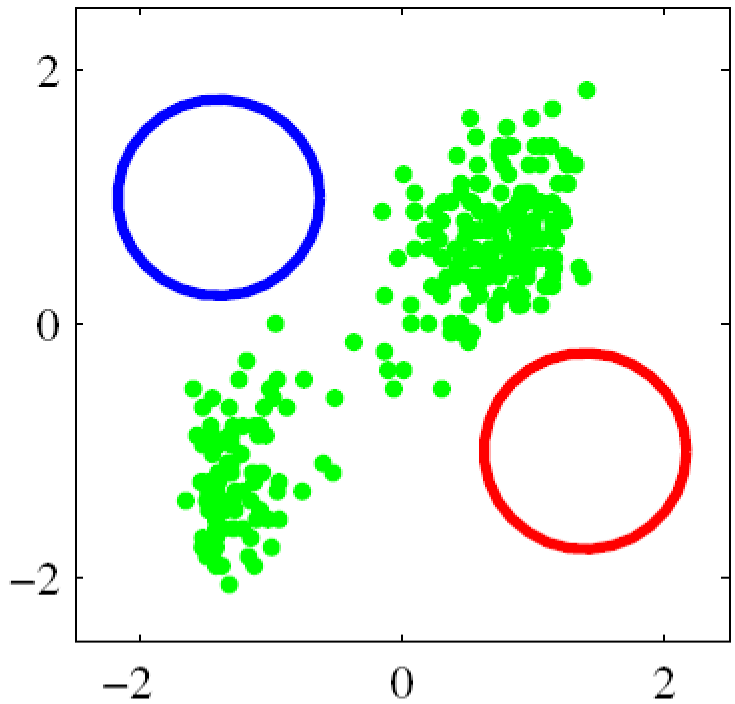
$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top$$

$$\pi_k = \frac{N_k}{N}$$



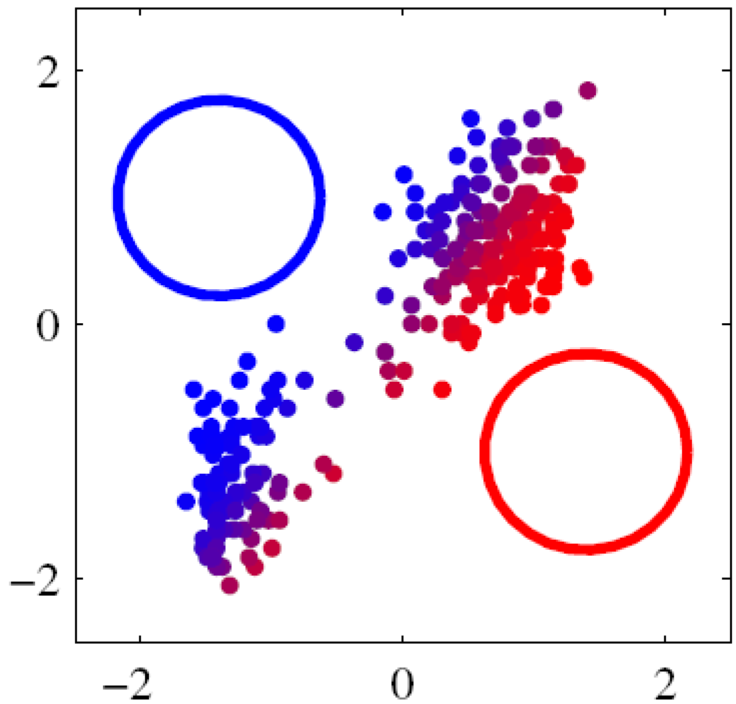
Initialization plays a key role to succeed!

A Running Example



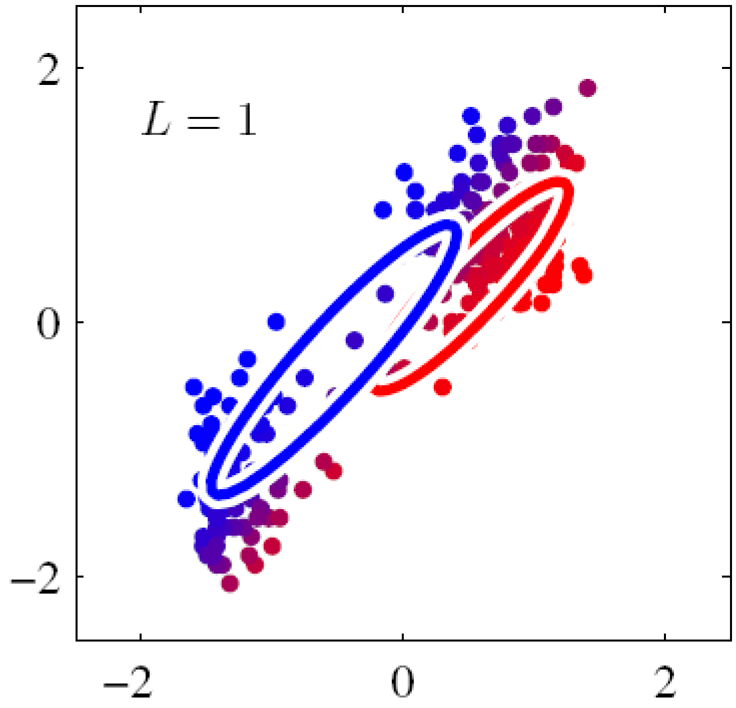
- The data and a mixture of two isotropic Gaussians

A Running Example



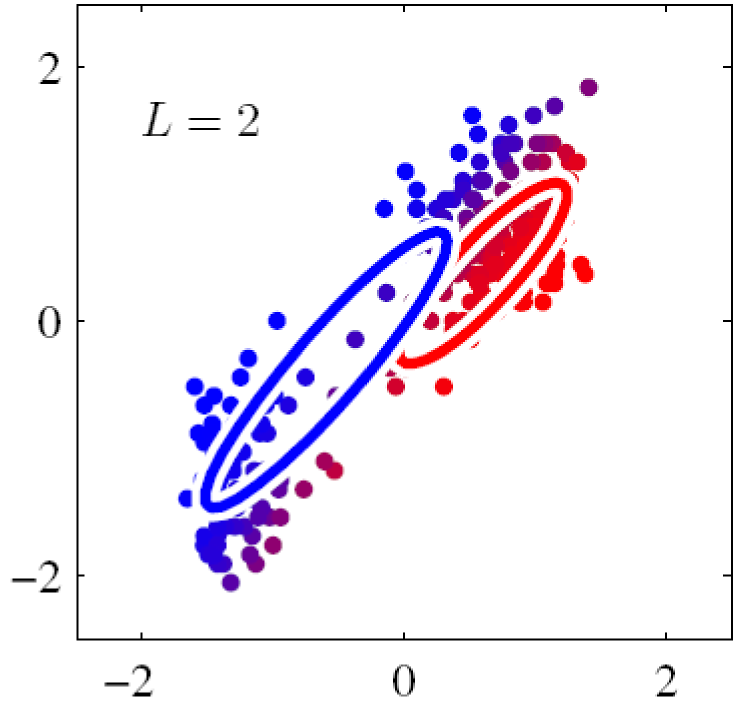
- Initial E-step

A Running Example



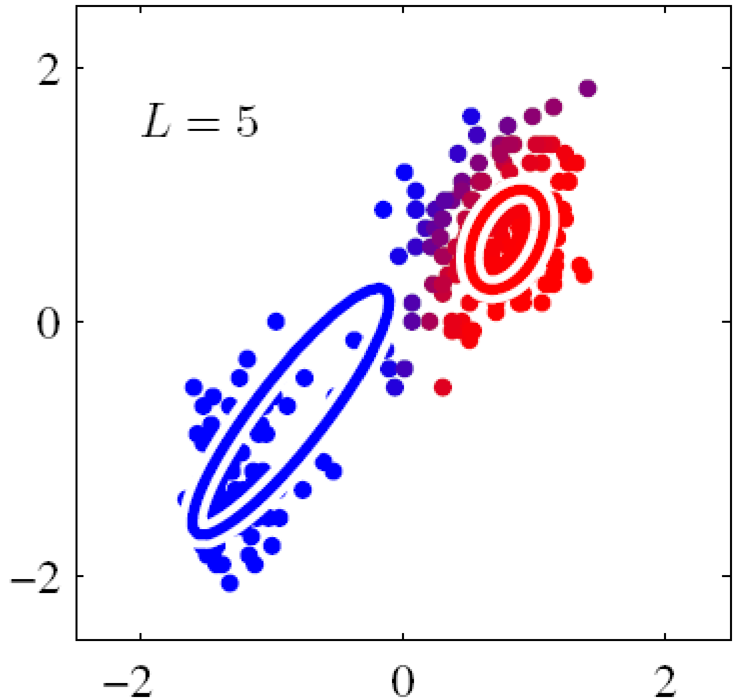
- Initial M-step

A Running Example



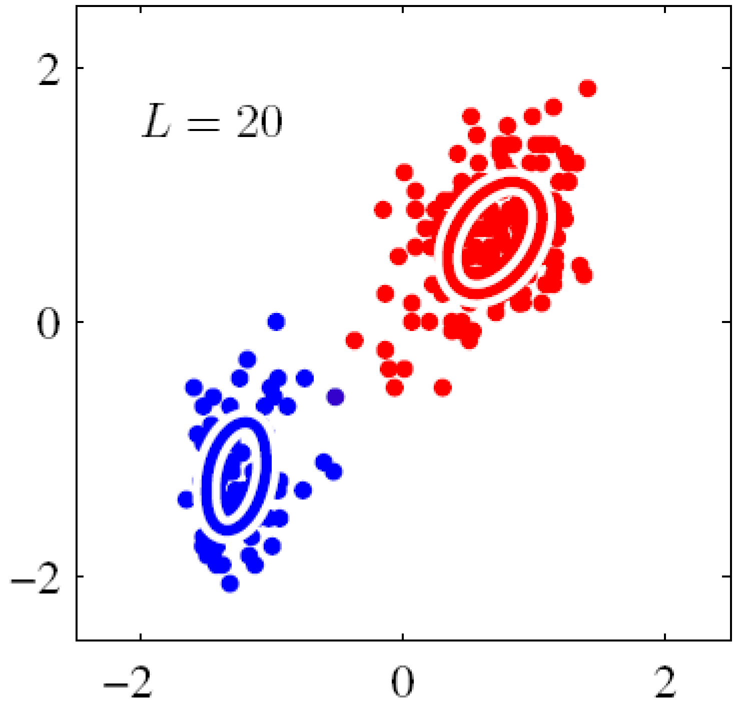
- The 2nd M-step

A Running Example



- The 5th M-step

A Running Example

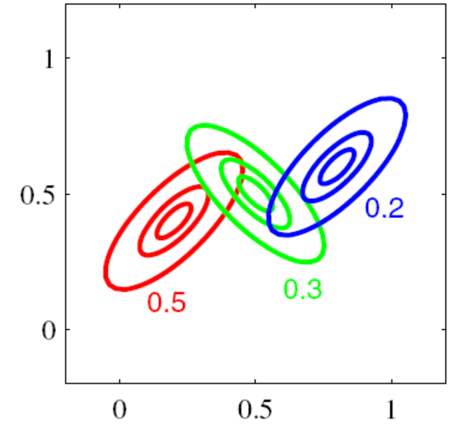


- The 20th M-step

Theory

- Let's take the latent variable view of mixture of Gaussians

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)$$

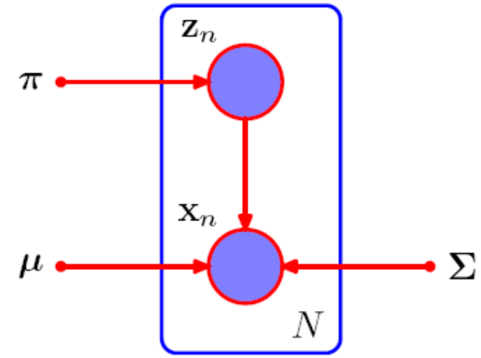


  - Indicator (selecting) variable

$$\mathbf{z} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

$$p(\mathbf{x}, \mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)^{z_k}$$

$$p(\mathbf{x}) \overset{?}{=} \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z})$$



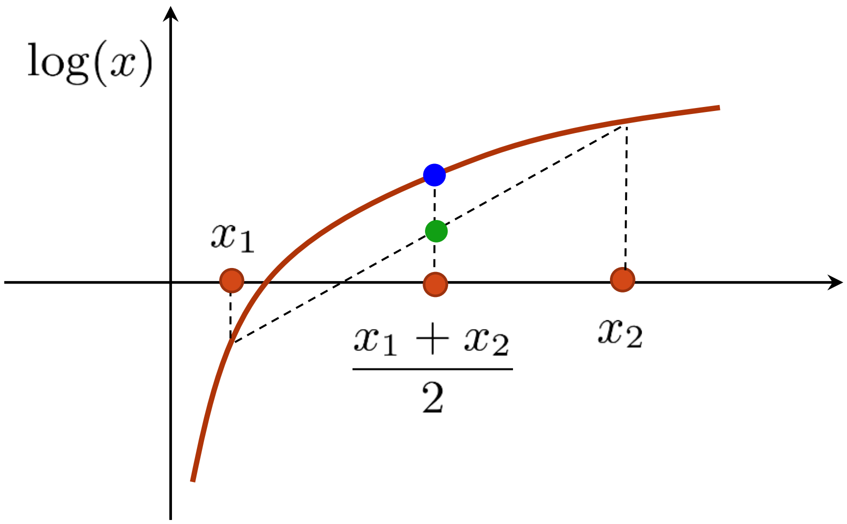
Note: the idea of data augmentation is influential in statistics and machine learning!

Theory

- Re-visit the log-likelihood

$$\log p(\mathcal{D}|\Theta) = \sum_{n=1}^{N} \log \left(\sum_{\mathbf{z}_n} p(\mathbf{x}_n, \mathbf{z}_n) \right)$$

- Jensen's inequality



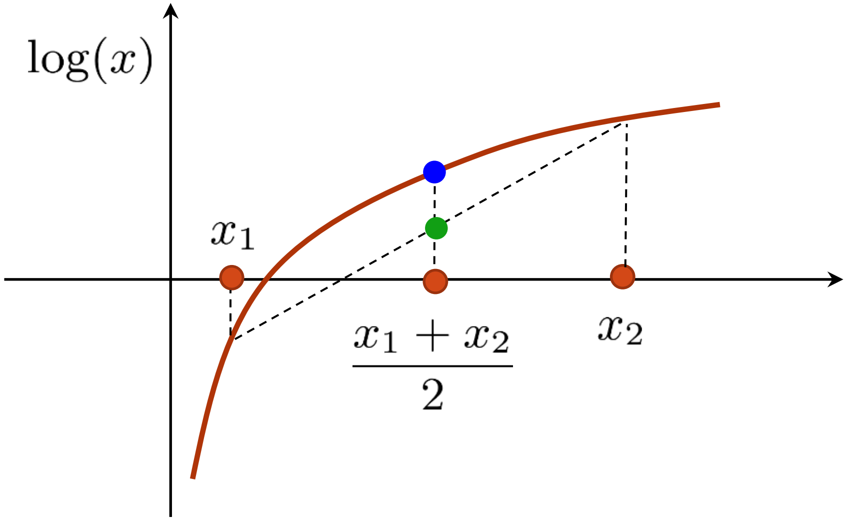
$$\boxed{\log \frac{x_1 + x_2}{2} \geq \frac{\log x_1 + \log x_2}{2}}$$

Theory

- Re-visit the log-likelihood

$$\log p(\mathcal{D}|\Theta) = \sum_{n=1}^{N} \log\left(\sum_{\mathbf{z}_n} p(\mathbf{x}_n, \mathbf{z}_n)\right)$$

- Jensen's inequality



$$\boxed{\log \mathbb{E}_{p(x)}[x] \geq \mathbb{E}_{p(x)}[\log x]}$$

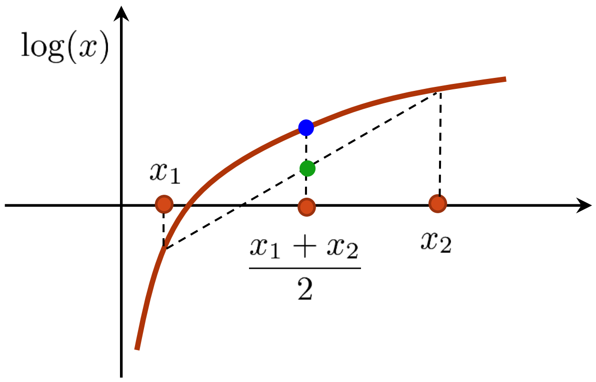
Theory

- Re-visit the log-likelihood

$$\log p(\mathcal{D}|\Theta) = \sum_{n=1}^{N} \log \Big(\sum_{\mathbf{z}_n} p(\mathbf{x}_n, \mathbf{z}_n) \Big)$$

- Jensen's inequality

$$\boxed{\log \mathbb{E}_{p(x)}[x] \geq \mathbb{E}_{p(x)}[\log x]}$$



- How to apply?

$$\log p(\mathcal{D}|\Theta) = \sum_{n=1}^{N} \log \Big(\sum_{\mathbf{z}_n} q(\mathbf{z}_n) \frac{p(\mathbf{x}_n, \mathbf{z}_n)}{q(\mathbf{z}_n)} \Big)$$

$$\geq \sum_{n=1}^{N} \sum_{\mathbf{z}_n} q(\mathbf{z}_n) \log \Big(\frac{p(\mathbf{x}_n, \mathbf{z}_n)}{q(\mathbf{z}_n)} \Big)$$

Theory

- What we have is a lower bound

$$\log p(\mathcal{D}|\Theta) \geq \sum_{n=1}^{N} \sum_{\mathbf{z}_n} q(\mathbf{z}_n) \log \left(\frac{p(\mathbf{x}_n, \mathbf{z}_n)}{q(\mathbf{z}_n)} \right) \triangleq \mathcal{L}(\Theta, q(\mathbf{Z}))$$

- What's the GAP?

$$\begin{aligned}
\mathcal{L}(\Theta, q(\mathbf{Z})) &= \sum_{n=1}^{N} \left\{ \sum_{\mathbf{z}_n} q(\mathbf{z}_n) \log p(\mathbf{x}_n, \mathbf{z}_n) - \sum_{\mathbf{z}_n} q(\mathbf{z}_n) \log q(\mathbf{z}_n) \right\} \\
&= \sum_{n=1}^{N} \left\{ \sum_{\mathbf{z}_n} q(\mathbf{z}_n) \log \left(\frac{p(\mathbf{x}_n, \mathbf{z}_n)}{p(\mathbf{x}_n)} \right) + \log p(\mathbf{x}_n) - \sum_{\mathbf{z}_n} q(\mathbf{z}_n) \log q(\mathbf{z}_n) \right\} \\
&= \log p(\mathcal{D}|\Theta) + \sum_{n=1}^{N} \left\{ \sum_{\mathbf{z}_n} q(\mathbf{z}_n) \log p(\mathbf{z}_n|\mathbf{x}_n) - \sum_{\mathbf{z}_n} q(\mathbf{z}_n) \log q(\mathbf{z}_n) \right\} \\
&= \log p(\mathcal{D}|\Theta) - \mathrm{KL}(q(\mathbf{Z}) \| p(\mathbf{Z}|\mathcal{D}))
\end{aligned}$$

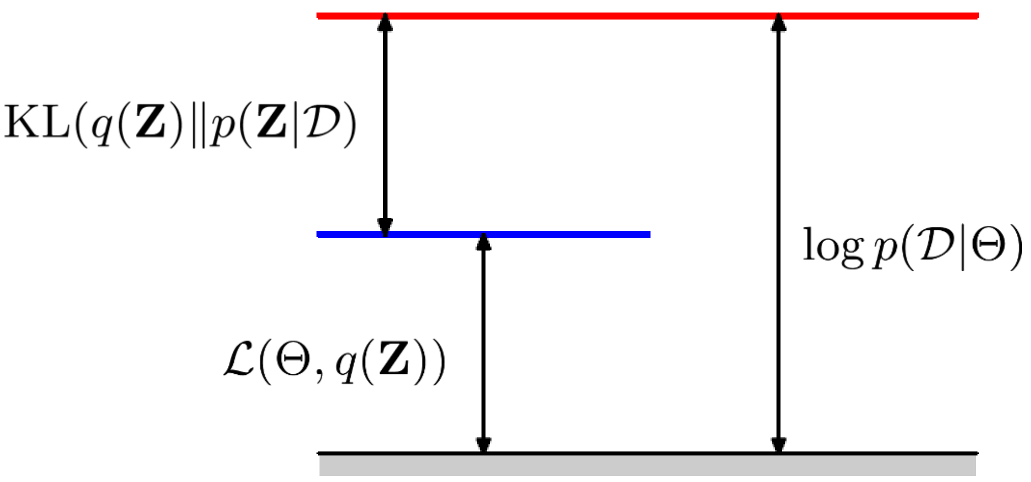
Theory

- What we have is a lower bound

$$\log p(\mathcal{D}|\Theta) \geq \sum_{n=1}^{N} \sum_{\mathbf{z}_n} q(\mathbf{z}_n) \log \left(\frac{p(\mathbf{x}_n, \mathbf{z}_n)}{q(\mathbf{z}_n)} \right) \triangleq \mathcal{L}(\Theta, q(\mathbf{Z}))$$

- What's the GAP?

$$\log p(\mathcal{D}|\Theta) - \mathcal{L}(\Theta, q(\mathbf{Z})) = \mathrm{KL}(q(\mathbf{Z}) \| p(\mathbf{Z}|\mathcal{D}))$$

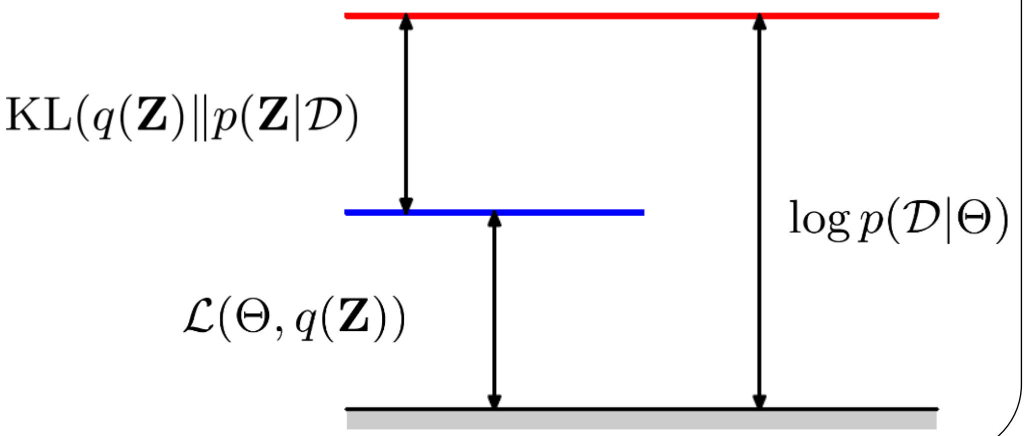


EM-algorithm

- Maximize the lower bound or minimize the gap:

$$\log p(\mathcal{D}|\Theta) \geq \sum_{n=1}^{N} \sum_{\mathbf{z}_n} q(\mathbf{z}_n) \log \left(\frac{p(\mathbf{x}_n, \mathbf{z}_n)}{q(\mathbf{z}_n)} \right) \triangleq \mathcal{L}(\Theta, q(\mathbf{Z}))$$

  - Maximize over q(Z) => E-step
  - Maximize over Θ => M-step



Convergence of EM

- Local optimum is guaranteed under mild conditions (Depster et al., 1977)
  - alternating minimization for a bi-convex problem

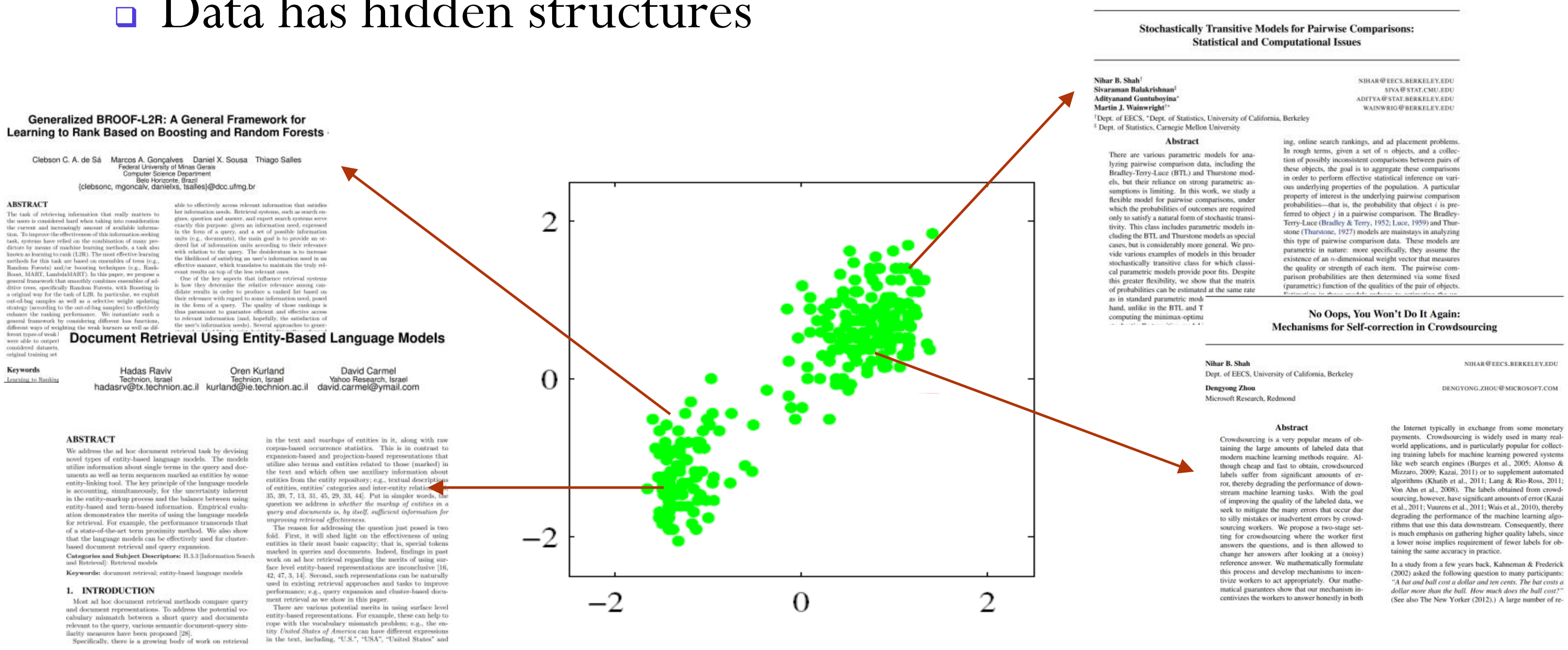
$$\mathcal{L}(\Theta_{t+1}) \geq \mathcal{L}(\Theta_t)$$

- Some special cases with global optimum (Wu, 1983)

- First-order gradient descent for log-likelihood
  - for comparison with other gradient ascent methods, see (Xu & Jordan, 1995)

Language model revisited

- A fully-observed model is not sufficient
  - Data has hidden structures



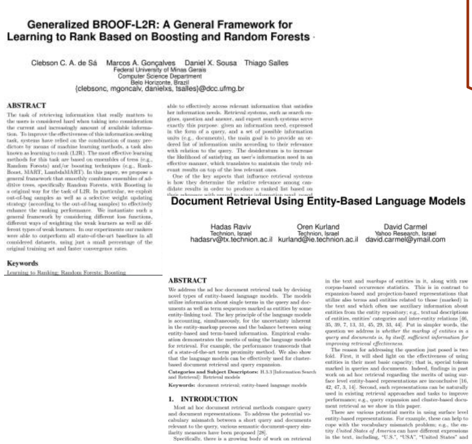
  - A simple distribution is not sufficient

Generative models are everywhere …

- Mixture model --- a simple generative model with hidden factors
  - Separate the data into different groups

T1=“SIGIR”

retrieval	0.05
text	0.02
learning	0.01
sports	0.01
…	


Generalized BROOF-L2R: A General Framework for Learning to Rank Based on Boosting and Random Forests

Document Retrieval Using Entity-Based Language Models

T2=“ICML”

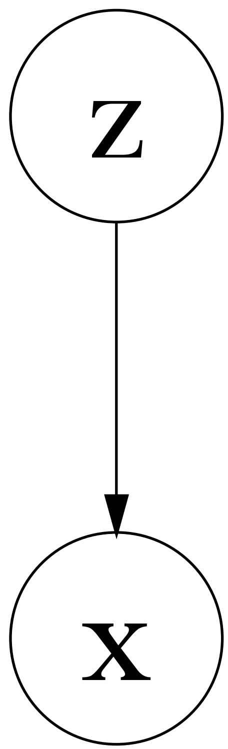
learning	0.03
theory	0.02
algorithm	0.01
deep	0.01
…	


Stochastically Transitive Models for Pairwise Comparisons: Statistical and Computational Issues

No Oops, You Won't Do It Again: Mechanisms for Self-correction in Crowdsourcing

Generative models are everywhere …

- Mixture model --- a simple generative model with hidden factors
  - Graphical model representation



$p(z)$

$p(x|z;\theta)$

$$p(x,z) = p(z)p(x|z;\theta)$$

Generalized BROOF-L2R: A General Framework for Learning to Rank Based on Boosting and Random Forests

Clebson C. A. de Sá Marcos A. Gonçalves Daniel X. Sousa Thiago Salles
Federal University of Minas Gerais
Computer Science Department
Belo Horizonte, Brazil
{clebsonc, mgoncalv, danielxs, tsalles}@dcc.ufmg.br

Document Retrieval Using Entity-Based Language Models

Hadas Raviv, Technion, Israel, hadasrv@tx.technion.ac.il
Oren Kurland, Technion, Israel, kurland@ie.technion.ac.il
David Carmel, Yahoo Research, Israel, david.carmel@ymail.com

Stochastically Transitive Models for Pairwise Comparisons: Statistical and Computational Issues

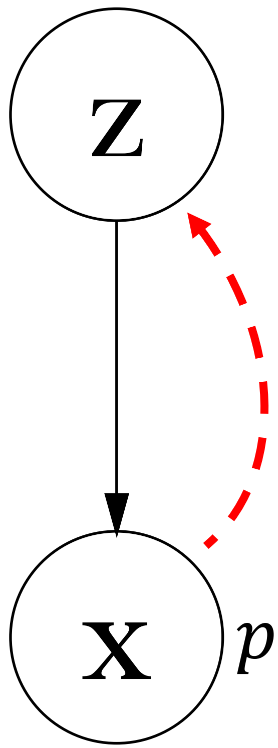
Nihar B. Shah, Sivaraman Balakrishnan, Adityanand Guntuboyina, Martin J. Wainwright

No Oops, You Won't Do It Again: Mechanisms for Self-correction in Crowdsourcing

Nihar B. Shah, Dept. of EECS, University of California, Berkeley
Dengyong Zhou, Microsoft Research, Redmond

Generative models are everywhere …

- Mixture model --- a simple generative model with hidden factors
  - Infer the latent Z:



$p(z)$

$p(x|z;\theta)$

Bayes' Rule:

$$p(z|x) = \frac{p(x,z)}{p(x)}$$

$$\propto p(z)p(x|z;\theta)$$

No Oops, You Won't Do It Again: Mechanisms for Self-correction in Crowdsourcing

Nihar B. Shah NIHAR@EECS.BERKELEY.EDU
Dept. of EECS, University of California, Berkeley

Dengyong Zhou DENGYONG.ZHOU@MICROSOFT.COM
Microsoft Research, Redmond

Abstract

Crowdsourcing is a very popular means of obtaining the large amounts of labeled data that modern machine learning methods require. Although cheap and fast to obtain, crowdsourced labels suffer from significant amounts of error, thereby degrading the performance of downstream machine learning tasks. With the goal of improving the quality of the labeled data, we seek to mitigate the many errors that occur due to silly mistakes or inadvertent errors by crowdsourcing workers. We propose a two-stage setting for crowdsourcing where the worker first answers the questions, and is then allowed to change her answers after looking at a (noisy) reference answer. We mathematically formulate this process and develop mechanisms to incentivize workers to act appropriately. Our mathematical guarantees show that our mechanism incentivizes the workers to answer honestly in both

the Internet typically in exchange from some monetary payments. Crowdsourcing is widely used in many real-world applications, and is particularly popular for collecting training labels for machine learning powered systems like web search engines (Burges et al., 2005; Alonso & Mizzaro, 2009; Kazai, 2011) or to supplement automated algorithms (Khatib et al., 2011; Lang & Rio-Ross, 2011; Von Ahn et al., 2008). The labels obtained from crowdsourcing, however, have significant amounts of error (Kazai et al., 2011; Vuurens et al., 2011; Wais et al., 2010), thereby degrading the performance of the machine learning algorithms that use this data downstream. Consequently, there is much emphasis on gathering higher quality labels, since a lower noise implies requirement of fewer labels for obtaining the same accuracy in practice.

In a study from a few years back, Kahneman & Frederick (2002) asked the following question to many participants: *"A bat and ball cost a dollar and ten cents. The bat costs a dollar more than the ball. How much does the ball cost?"* (See also The New Yorker (2012).) A large number of re-

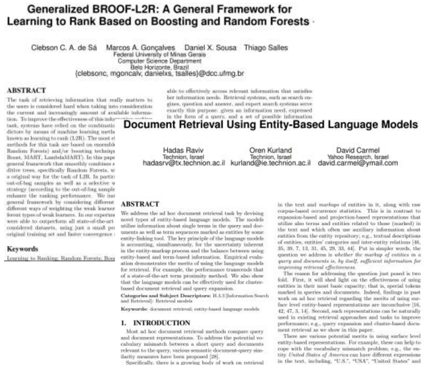
Generative models are everywhere …

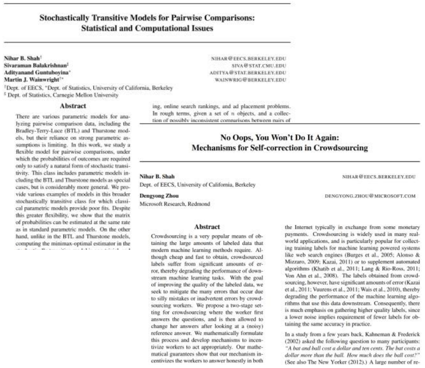
- Mixture model --- a simple generative model with hidden factors
  - EM algorithm to learn the unknown language models

E-step: Infer the hidden Z

M-step: Update the parameters

T1=“SIGIR”   T2=“ICML”





retrieval 0.05
text 0.02
learning 0.01
sports 0.01
…

learning 0.03
theory 0.02
algorithm 0.01
deep 0.01
…

Thanks!

ZhuSuan: A Library for Bayesian Deep Learning. J. Shi, J. Chen, J. Zhu, S. Sun, Y. Luo, Y. Gu, Y. Zhou. arXiv preprint, arXiv:1709.05870 , 2017

Online Documents: http://zhusuan.readthedocs.io/