Lecture 14: Generative Classifiers and Mixture Models

TTIC 31020: Introduction to Machine Learning

Instructor: Kevin Gimpel

TTI-Chicago

November 14, 2019

Classification and zero/one loss

- Want to minimize the zero/one loss for classifier $h : \mathcal{X} \to \mathcal{Y}$, which for $(\mathbf{x}, y)$ is

$$\ell(h(\mathbf{x}), y) = \begin{cases} 0 & \text{if } h(\mathbf{x}) = y, \\ 1 & \text{if } h(\mathbf{x}) \neq y. \end{cases}$$

Risk of a classifier

- The risk (expected loss) of a C-way classifier $h(\mathbf{x})$:

$$
\begin{aligned}
R(h) &= \mathbb{E}_{\mathbf{x},y}\left[\ell(h(\mathbf{x}), y)\right] \\
&= \int_{\mathbf{x}} \sum_{c=1}^{C} \ell(h(\mathbf{x}), c)\, p(\mathbf{x}, y=c)\, d\mathbf{x} \\
&= \int_{\mathbf{x}} \left[\sum_{c=1}^{C} \ell(h(\mathbf{x}), c)\, p\left(y=c \,|\, \mathbf{x}\right)\right] p(\mathbf{x}) d\mathbf{x}
\end{aligned}
$$

- It's enough to minimize the **conditional risk** for any $\mathbf{x}$ (there are no restrictions on h):

$$
R(h \,|\, \mathbf{x}) = \sum_{c=1}^{C} \ell(h(\mathbf{x}), c) p\left(y=c \,|\, \mathbf{x}\right)
$$

Conditional risk of a classifier

$$
\begin{aligned}
R(h \mid \mathbf{x}) &= \sum_{c=1}^{C} \ell(h(\mathbf{x}), c) p\left(y = c \mid \mathbf{x}\right) \\
&= 0 \cdot p\left(y = h(\mathbf{x}) \mid \mathbf{x}\right) + 1 \cdot \sum_{c \neq h(\mathbf{x})} p\left(y = c \mid \mathbf{x}\right) \\
&= \sum_{c \neq h(\mathbf{x})} p\left(y = c \mid \mathbf{x}\right) = 1 - p\left(y = h(\mathbf{x}) \mid \mathbf{x}\right)
\end{aligned}
$$

- To minimize conditional risk given $\mathbf{x}$, the classifier should be

$$h(\mathbf{x}) = \operatorname*{argmax}_{c} p\left(y = c \mid \mathbf{x}\right)$$

- This is the *best possible* classifier in terms of generalization, i.e., expected misclassification rate on new examples

Optimal classification

- Expected classification error is minimized by

$$h(\mathbf{x}) = \operatorname*{argmax}_c p\,(y = c \,|\, \mathbf{x})$$

- Definition: **Bayes classifier** minimizes expected classification error:

$$h^*(\mathbf{x}) \;=\; \operatorname*{argmax}_c p\,(y = c \,|\, \mathbf{x})$$

- Using Bayes rule, we can rewrite the Bayes classifier as follows:

$$\begin{aligned} h^*(\mathbf{x}) &= \operatorname*{argmax}_c p\,(y = c \,|\, \mathbf{x}) \\ &= \operatorname*{argmax}_c \frac{p\,(\mathbf{x} \,|\, y = c)\, p(y = c)}{p(\mathbf{x})} \\ &= \operatorname*{argmax}_c p\,(\mathbf{x} \,|\, y = c)\, p(y = c) \end{aligned}$$

- This final line is often called a **generative classifier**

Discriminative and generative classifiers

- Discriminative:

$$\operatorname*{argmax}_c p\,(y = c \,|\, \mathbf{x})$$

  - Learn parameters of $p\,(y = c \,|\, \mathbf{x})$ (e.g., by minimizing log loss)
  - No need to learn distribution over $\mathbf{x}$
  - Usually better held-out accuracy (when there's enough training data and train/test data drawn from same distribution)

Discriminative and generative classifiers

- Discriminative:

$$\underset{c}{\operatorname{argmax}}\, p\,(y = c \,|\, \mathbf{x})$$

- Generative:

$$\underset{c}{\operatorname{argmax}}\, p\,(\mathbf{x} \,|\, y = c)\, p(y = c)$$

  - Instead of learning $p\,(y = c \,|\, \mathbf{x})$, learn $p\,(\mathbf{x} \,|\, y = c)$ and $p(y = c)$
  - We often use simple parametric models for $p\,(\mathbf{x} \,|\, y = c)$: multivariate Gaussians for continuous data, simple categorical/multinomial distributions for discrete data (today)
  - Even if we do a poor job estimating $p\,(\mathbf{x} \,|\, y = c)$, we may still get good classification accuracy
  - Generative classifiers can outperform discriminative when training sets are small or under other challenging conditions (e.g., noisy data)

Bayes risk

- The Bayes classifier h^* minimizes the expected classification error
- The risk (probability of error) of the Bayes classifier h^* is called the **Bayes risk** R^*
- This is the *minimal achievable* risk for the true distribution $p(\mathbf{x}, y)$ with any classifier!
- In a sense, R^* measures the inherent difficulty of the classification problem.

$$R^* \; = 1 - \int_{\mathbf{x}} \max_c \left(p\left(y = c \,|\, \mathbf{x}\right)\right) p(\mathbf{x}) d\mathbf{x}$$

- When is R^* equal to 0? True distribution $p(y \mid \mathbf{x})$ has probability 1 for a single class and 0 for all other classes

Review: discriminant functions

- We can construct, for each class c, a **discriminant function**

$$\delta_c(\mathbf{x}) \triangleq \log p\left(\mathbf{x} \mid y = c\right) + \log p(y = c)$$

max a posterior

such that optimal (Bayes) classifier is

$$h^*(\mathbf{x}) = \operatorname*{argmax}_c \delta_c(\mathbf{x})$$

- Intuition: choose c if $p\left(\mathbf{x} \mid y = c\right)$ explains $\mathbf{x}$ better, adjusted for the prior $p(y = c)$
- Can simplify δ_c by removing terms and factors common for all δ_c since they won't affect the decision boundary

Review: two-category case

- In case of two classes $y \in \{\pm 1\}$, the Bayes classifier is

$$h^*(\mathbf{x}) \;=\; \operatorname*{argmax}_{c=\pm 1} \delta_c(\mathbf{x}) \;= \operatorname{sign}\left(\delta_{+1}(\mathbf{x}) - \delta_{-1}(\mathbf{x})\right)$$

which is bigger

- Decision boundary is given by $\delta_{+1}(\mathbf{x}) - \delta_{-1}(\mathbf{x}) = 0$

Equal covariance Gaussian case

- Consider the case where $\mathbf{x} \mid c \sim \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma})$ ("$\mathbf{x}$ conditioned on c follows a normal distribution with parameters $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}$"), and equal prior for all classes.

$$
\begin{aligned}
\delta_k(x) &= \log p(\mathbf{x} \mid y = k) \\
&= \underbrace{-\log(2\pi)^{d/2} - \frac{1}{2}\log(|\boldsymbol{\Sigma}|)}_{\text{same for all } k} - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) \\
&\propto \text{const} - \underbrace{\mathbf{x}^\top \boldsymbol{\Sigma}^{-1} \mathbf{x}}_{\text{same for all } k} + \boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}^{-1} \mathbf{x} + \mathbf{x}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k - \boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k
\end{aligned}
$$

- Now consider two classes r and q:

$$
\begin{aligned}
\delta_r(\mathbf{x}) &\propto 2\boldsymbol{\mu}_r^\top \boldsymbol{\Sigma}^{-1} \mathbf{x} - \boldsymbol{\mu}_r^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_r \\
\delta_q(\mathbf{x}) &\propto 2\boldsymbol{\mu}_q^\top \boldsymbol{\Sigma}^{-1} \mathbf{x} - \boldsymbol{\mu}_q^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_q
\end{aligned}
$$

Note: combining terms above is possible because a Gaussian covariance matrix is symmetric and invertible, and the inverse of a symmetric matrix is symmetric

Linear discriminant

- Two-class discriminants (contest between two classes):

$$\begin{aligned} \delta_r(\mathbf{x}) - \delta_q(\mathbf{x}) &= 2\boldsymbol{\mu}_r^\top \boldsymbol{\Sigma}^{-1}\mathbf{x} - \boldsymbol{\mu}_r^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_r \\ &- \left(2\boldsymbol{\mu}_q^\top \boldsymbol{\Sigma}^{-1}\mathbf{x} - \boldsymbol{\mu}_q^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_q\right) \\ &= \mathbf{w} \cdot \mathbf{x} + b \end{aligned}$$

- Decision boundary can be written as a linear function of $\mathbf{x}$!

same sigma

Parameters in Gaussian ML

- When using Gaussian distributions for $p(\mathbf{x} \mid y = c)$, how many parameters do we need to learn for each class?
- Single Gaussian in $\mathbb{R}^d$: d for the mean, plus

$\boldsymbol{\Sigma} =$	$\begin{bmatrix} \sigma_1^2 & \sigma_{12} & \dots & \sigma_{1d} \\ \sigma_{12} & \sigma_2^2 & \dots & \sigma_{2d} \\ \cdot & \cdot & \cdot & \cdot \\ \sigma_{1d} & \cdot & \cdot & \sigma_d^2 \end{bmatrix}$	$\begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \sigma_d^2 \end{bmatrix}$	$\sigma^2\mathbf{I}$
# param	$d + d(d-1)/2$	d	1

- Diagonal covariance means effectively assuming feature independence
- Spherical covariance means independence *and* equal variance in all directions

Naïve Bayes (NB) classifier

- Suppose $\mathbf{x}$ is represented by m features $\phi_1(\mathbf{x}), \ldots, \phi_m(\mathbf{x})$
- NB assumes the features are conditionally independent given the class:

$$p\,(\mathbf{x} \,|\, y = c) \;=\; p\,(\phi_1(\mathbf{x}), \ldots, \phi_m(\mathbf{x}) \,|\, y = c) \;=\; \prod_{j=1}^{m} p\,(\phi_j(\mathbf{x}) \,|\, y = c)$$

- This is typically an oversimplification of the data (hence the name "naïve") but often works pretty well in practice
- Under this assumption, the Bayes classifier is

$$h^*(\mathbf{x}) = \operatorname{sign}\left[\sum_{j=1}^{m} \log \frac{p\,(\phi_j(\mathbf{x}) \,|\, y = +1)}{p\,(\phi_j(\mathbf{x}) \,|\, y = -1)} + \underbrace{\log p(y=+1) - \log p(y=-1)}_{\text{(log)prior over classes}}\right]$$

compare posterior of +1 and -1 classes

Naïve Bayes for Gaussian model

$$p(\mathbf{x} \mid y=c) \;=\; p(\phi_1(\mathbf{x}), \ldots, \phi_m(\mathbf{x}) \mid y=c) \;=\; \prod_{j=1}^{m} p(\phi_j(\mathbf{x}) \mid y=c)$$

- Assume each feature follows a class-conditional Gaussian distribution:

$$\phi_j(\mathbf{x}) \mid c \sim \mathcal{N}(\mu_{jc}, \sigma^2_{jc})$$

  NB assumption of independence is equivalent to

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma^2_{1c} & 0 & \ldots & 0 \\ 0 & \sigma^2_{2c} & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \ldots & \sigma^2_{mc} \end{bmatrix}$$

- Need to estimate m 1D Gaussian densities for each class.

Example: generative classifiers for documents

- A common task: given an e-mail message, classify it as SPAM ($y = 1$) or "ham" (a legitimate e-mail, $y = 0$).
- Define a set of keywords $W_1, \ldots, W_m$. Then,

$$\phi_j(\mathbf{x}) = \begin{cases} 1 & \text{document } \mathbf{x} \text{ includes } W_j \\ 0 & \text{otherwise} \end{cases}$$

- A document $\mathbf{x}$ (of arbitrary length!) is now represented as a vector in $\{0,1\}^m$: $\Phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_m(\mathbf{x})]^\top$
- A natural distribution for $\phi_j(\mathbf{x})$ is **Bernoulli**: $p(\phi_j(\mathbf{x}) = 1; \theta_j) = \theta_j$
- We have two classes, so we have two class-conditional Bernoulli distributions for each feature (for simplicity, we write ϕ_j instead of $\phi_j(\mathbf{x})$):

$$p\left(\phi_j \,|\, y = 1\right) = \theta_{j1}^{\phi_j}(1 - \theta_{j1})^{1-\phi_j}$$

$$p\left(\phi_j \,|\, y = 0\right) = \theta_{j0}^{\phi_j}(1 - \theta_{j0})^{1-\phi_j}$$

Application: SPAM detection

- Given an email, classify it as SPAM ($y = 1$) or "ham" ($y = 0$), based on the content
- An important problem!
- Typical binary features:
  - keywords
  - HTML tags and patterns
  - TEXT IN ALL CAPS
  - number of recipients above certain threshold
  - comes from "blacklisted" address
  - etc.

SPAM detection with Naïve Bayes

- For simplicity, we will write ϕ_j instead of $\phi_j(\mathbf{x})$
- For a single binary feature ϕ_j,

$$p(\phi_j \mid y = 1) = \theta_{j1}^{\phi_j}(1 - \theta_{j1})^{1-\phi_j}$$

$$p(\phi_j \mid y = 0) = \theta_{j0}^{\phi_j}(1 - \theta_{j0})^{1-\phi_j}$$

- ML estimate of a Bernoulli variable:
  k SPAM emails with $\phi_j = 1$ and $n - k$ with $\phi_j = 0 \Rightarrow \hat{\theta}_{j1} = k/n$.

Classifying a document

- Given new document $\mathbf{x} = [\phi_1, \ldots, \phi_m]^\top$:

2m theta

and 2 prior

$$\hat{y} = 1 \Leftrightarrow \sum_{j=1}^{m} \phi_j \log \theta_{j1} + \sum_{j=1}^{m} (1 - \phi_j) \log(1 - \theta_{j1}) \\ - \sum_{j=1}^{m} \phi_j \log \theta_{j0} - \sum_{j=1}^{m} (1 - \phi_j) \log(1 - \theta_{j0}) \\ + \log p(y = 1) - \log p(y = 0) \ \geq \ 0$$

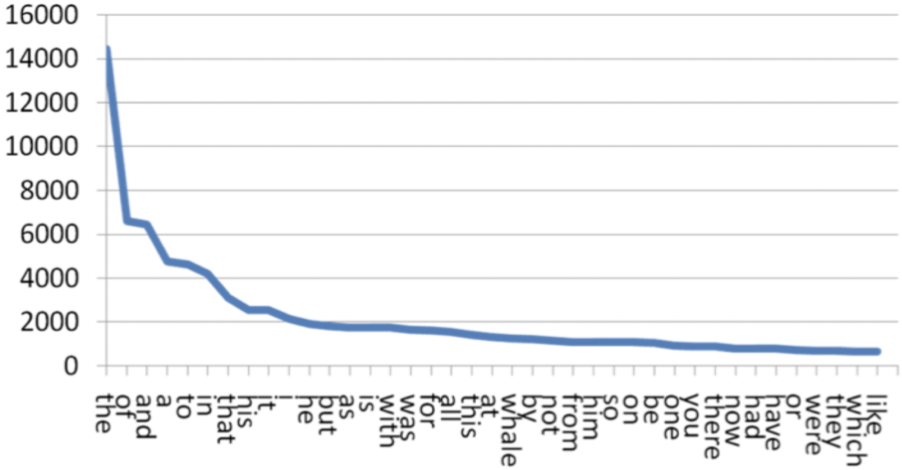
- There are a total of $2 + 2m$ parameters to estimate in this model

MLE is too sensitive to the data: for example, flip 5 coins, 5 heads then theta = 1, which is unlikely.

Problems with ML estimation

- Recall the coin-tossing experiments:
  - ML is too sensitive to the data, and may violate some "reasonable" beliefs about θ, e.g., that $\theta = 1$ is very unlikely.
- A real problem in text classification: *Zipf's law* for natural language: n-th most common word has relative frequency $1/n^a$, with $a \approx 1$.
  - relative frequency means $\#(\text{this word})/\#(\text{all words})$

Zipf's law



blog EMIS

Problems with ML estimation

- According to ML, when a word appears in a message that we have never seen in SPAM, we *must* predict it's non-SPAM (due to zero probabilities).
- If the same message contains a word never seen in non-SPAM, what do we do?

MAP estimate for word counts

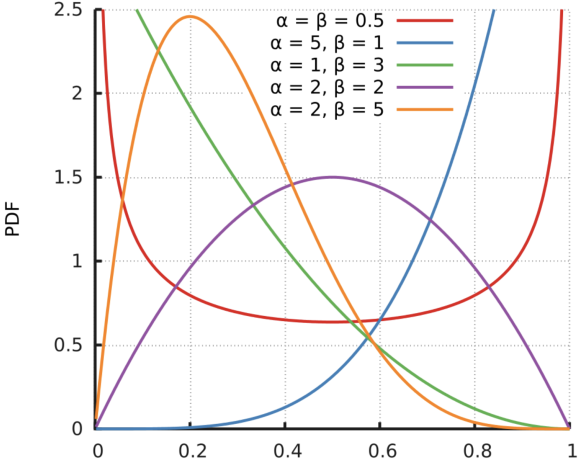
- We can use a Beta distribution as a **prior** on each parameter θ:

$$B(\theta;\, a, b) \;\triangleq\; \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\theta^{a-1}(1-\theta)^{b-1}$$

- The beta distribution assigns probabilities to values of θ and has two parameters of its own: a and b

Beta distribution

$$B(\theta;\, a, b) \;\triangleq\; \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\theta^{a-1}(1-\theta)^{b-1}$$



(Note: the plot shows α and β; just think of those as a and b, respectively)

MAP estimate for word counts

- We can use the Beta distribution as a **prior** on the parameter θ:

$$B(\theta;\, a, b) \;\triangleq\; \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\theta^{a-1}(1-\theta)^{b-1}$$

  Beta distribution is conjugate prior to Bernoulli likelihood:

$$p\,(\theta\,|\,X) \;\propto\; p(\theta;\, n_1 + a, n_0 + b)$$

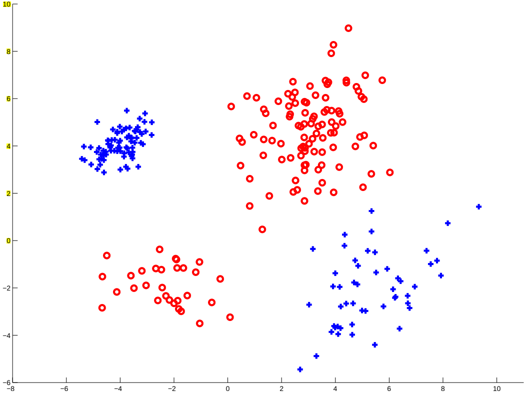
conjugate class of Bernoulli and Beta

- Interpretation: a and b are pseudocounts
  - Prior $p(\theta) = B(\theta; a, b)$ is equivalent to having seen $a + b$ observations, a of which were ones and b zeros, *before* we observed actual data X_n.
  - An alternative phrasing: $a/(a + b)$ is the default value for θ, and $a + b$ is how strongly we believe in that value.
- The posterior $p\,(\theta\,|\,X_n)$ updates that by adding the actual counts to the pseudocounts.

Mixture models

- So far, we have assumed that each class has a single coherent model.

we can choose a Gaussian mixture model here. But if we do not choose gaussian, there is not a closed form for p(X|Y=c). So no cost function but can use EM method to estimate



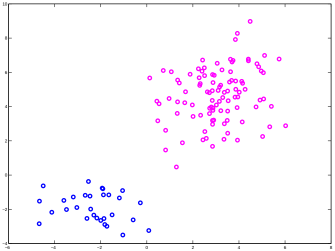
- What if the examples (within the same class) are from a number of distinct "types"?

Examples

- Images of the same person under different conditions: with/without glasses, different expressions, different views.
- Images of the same category but different sorts of objects: chairs with/without armrests.
- For document classification, multiple distinct subareas within a single class, e.g., in the class "sports", may have natural clusters of documents about baseball and football.
- Different ways of pronouncing the same phonemes in speech recognition.
- We may have intuitions that the data contains these sorts of clusters, but we may not know which examples belong to which cluster!
- Can we learn this automatically?

Mixture models

- Assumptions:
  - k underlying types (components);
  - y_i is the identity of the component "responsible" for $\mathbf{x}_i$;
  - y_i is a **hidden (latent)** variable: never observed.



- A **mixture model**:

$$p(\mathbf{x};\, \boldsymbol{\pi}) \;=\; \sum_{c=1}^{k} p(y=c)\, p\left(\mathbf{x} \,|\, y=c\right)$$

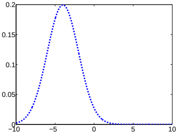
- $\pi_c \;\triangleq\; p(y=c)$ are the **mixing probabilities**
- We need to parameterize the component densities $p\left(\mathbf{x} \,|\, y=c\right)$.

Parametric mixtures

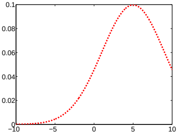
- Suppose that the parameters of the c-th component are θ_c. Then we can denote $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k]$ and write

$$p(\mathbf{x}; \boldsymbol{\theta}, \boldsymbol{\pi}) \; = \; \sum_{c=1}^{k} \pi_c \cdot p(\mathbf{x};\, \boldsymbol{\theta}_c)$$

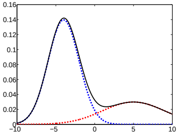
- Any valid setting of $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$, subject to $\sum_{c=1}^{k} \pi_c = 1$, produces a valid pdf.
- Example: mixture of Gaussians.



$\times 0.7 \quad + $



$\times 0.3 \quad = $



Generative model for a mixture

- The generative process with k-component mixture:
  - The parameters $\boldsymbol{\theta}_c$ for each component c are fixed.
  - Draw $y_i \sim [\pi_1, \ldots, \pi_k]$;
  - Given y_i, draw $\mathbf{x}_i \sim p\,(\mathbf{x} \,|\, y_i;\, \boldsymbol{\theta}_{y_i})$.
- The entire generative model for $\mathbf{x}$ and y:

$$p(\mathbf{x}, y;\, \boldsymbol{\theta}, \boldsymbol{\pi}) \;=\; p(y; \boldsymbol{\pi}) \cdot p(\mathbf{x}|y;\, \boldsymbol{\theta}_y)$$

- Any data point $\mathbf{x}_i$ could have been generated in k ways.
- If the c-th component is a Gaussian, $p\,(\mathbf{x} \,|\, y = c) = \mathcal{N}(\mathbf{x}; \underbrace{\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c}_{\boldsymbol{\theta}_c})$,

gaussian mixture

$$p(\mathbf{x};\, \theta, \boldsymbol{\pi}) \;=\; \sum_{c=1}^{k} \pi_c \cdot \mathcal{N}\left(\mathbf{x};\, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\right),$$

where $\theta = [\mu_1, \ldots, \mu_k, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_k]$.

Likelihood of a mixture model

- Idea: estimate set of parameters that maximize likelihood given the observed data.
- The log-likelihood of $\boldsymbol{\pi}, \boldsymbol{\theta}$:

$$\log p(X;\, \boldsymbol{\pi}, \theta) \;=\; \sum_{i=1}^{n} \log \sum_{c=1}^{k} \pi_c \, \mathcal{N}\left(\mathbf{x}_i;\, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\right).$$

- No closed-form solution because of the sum inside log.
  - To compute, we would need to take into account all possible components that could have generated $\mathbf{x}_i$.
  - k^n possible assignments!