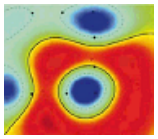
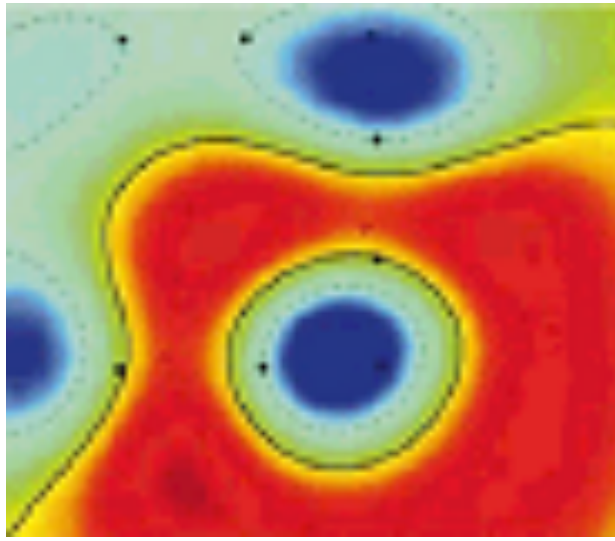


Technische Universität Berlin



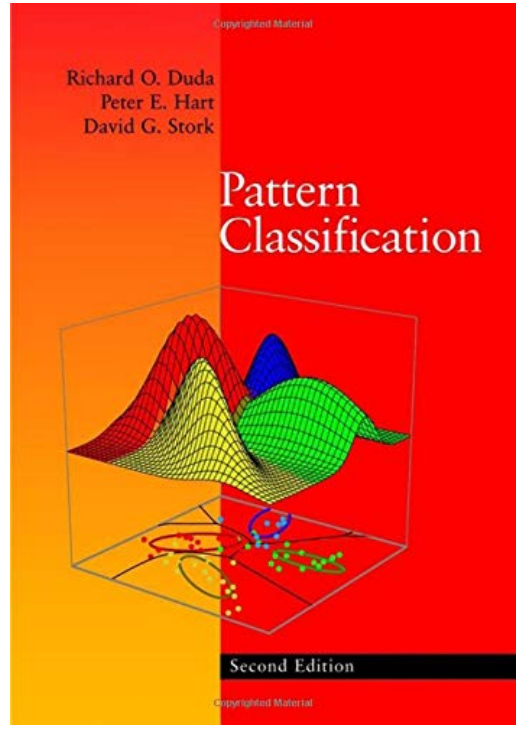
Lecture 2

Parameter Estimation

Textbook Reference

Duda et al. Pattern Classification, 2nd Edition (2000)

- This week: Sections 3.1–3.5



Recap: Bayes Decision Theory

Recap:

- Knowing class priors and class-conditioned data densities, we can infer posterior probabilities using the Bayes theorem

$$P(\omega_j \,|\, \boldsymbol{x}) = \frac{p(\boldsymbol{x} \,|\, \omega_j) \cdot P(\omega_j)}{p(\boldsymbol{x})}$$

- From there, optimal classifiers can be built, e.g. the maximum accuracy classifier:

$$\begin{aligned} &\arg\max_j \ \big[P(\omega_j \,|\, \boldsymbol{x})\big] \\ &= \arg\max_j \ \big[\log p(\boldsymbol{x} \,|\, \omega_j) + \log P(\omega_j)\big] \end{aligned}$$

Question:

- Can we assume that we know in advance the data densities $p(\boldsymbol{x} \,|\, \omega_j)$, or should they be learned from the data?

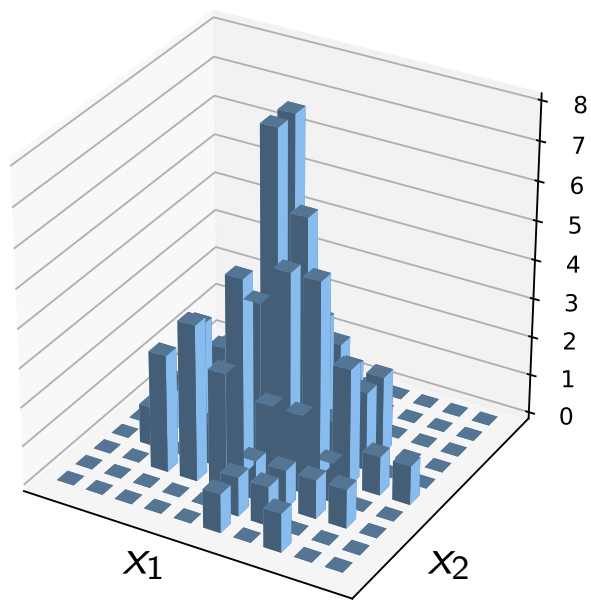
Learning $p(x \mid \omega_j)$: Histogram

Idea:

- Build a grid in input space, and for each class build a histogram that counts the number of observations that belong to each bin.

Problem:

- The number of bins is s^d where s is the number of steps along each dimension and d is the number of dimensions.
- Many bins will have zero observations, not because the data probability is truly zero, but because finitely many examples have been observed.



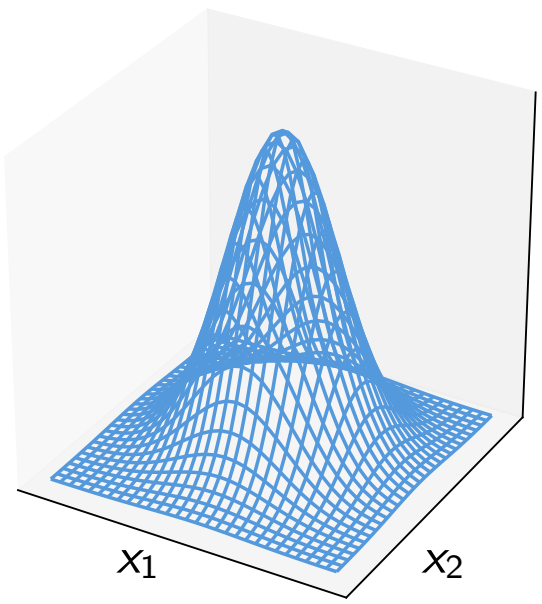
Learning $p(\boldsymbol{x} \mid \omega_j)$: Model-Based

Idea:

- Assume that $p(\boldsymbol{x} \mid \omega_j)$ is a known parametric form, e.g. $\mathcal{N}(\boldsymbol{\mu}_j, \Sigma_j)$ where $\boldsymbol{\mu}_j, \Sigma_j$ are the parameters of the distribution.
- Estimate the parameters of the distribution that best fit the few observations we have.

Advantage:

- With the model-based approach, we need to estimate a finite and typically small number of model parameters and not the whole data distribution.



Maximum Likelihood Estimation

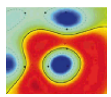
Goal: Let $\{p(\boldsymbol{x} \mid \theta), \theta \in \Theta\}$ be a set of density functions (e.g. Gaussian), where θ denotes a parameter vector (e.g. mean / covariance). We would like to find the parameter θ that is the most likely with respect to the data.

Maximum Likelihood (ML) Approach:

- Assume that we have a dataset $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$.
- Assume that each example $\boldsymbol{x}_k \in \mathbb{R}^d$ in the dataset has been is generated independently and from the same density function $p(\boldsymbol{x} \mid \theta)$.
- In that case, the joint density function can be written as:

$$\boxed{p(\mathcal{D} \mid \theta) = \prod_{k=1}^{N} p(\boldsymbol{x}_k \mid \theta)}$$

  We also call this quantity the *likelihood* of θ w.r.t. the dataset $\mathcal{D}$.
- The best parameter is then given by $\hat{\theta} = \arg\max_{\theta} p(\mathcal{D} \mid \theta)$.



Max Likelihood, Simple Gaussian Case

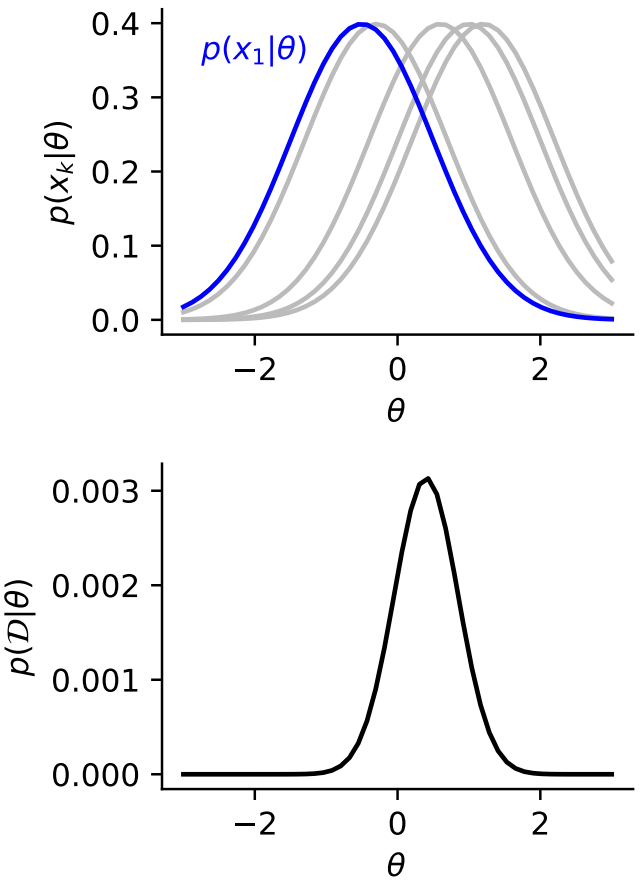
Assume the data density is modeled as a univariate Gaussian with unit variance and unknown mean θ. For a given data point x_k, the density function can be written as:

$$p(x_k \mid \theta) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x_k - \theta)^2\right]$$

Using the independence assumption, the joint density becomes:

$$p(\mathcal{D} \mid \theta) = \prod_{k=1}^{N} \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x_k - \theta)^2\right]$$

Question: How to find θ that maximizes the function $p(\mathcal{D} \mid \theta)$.



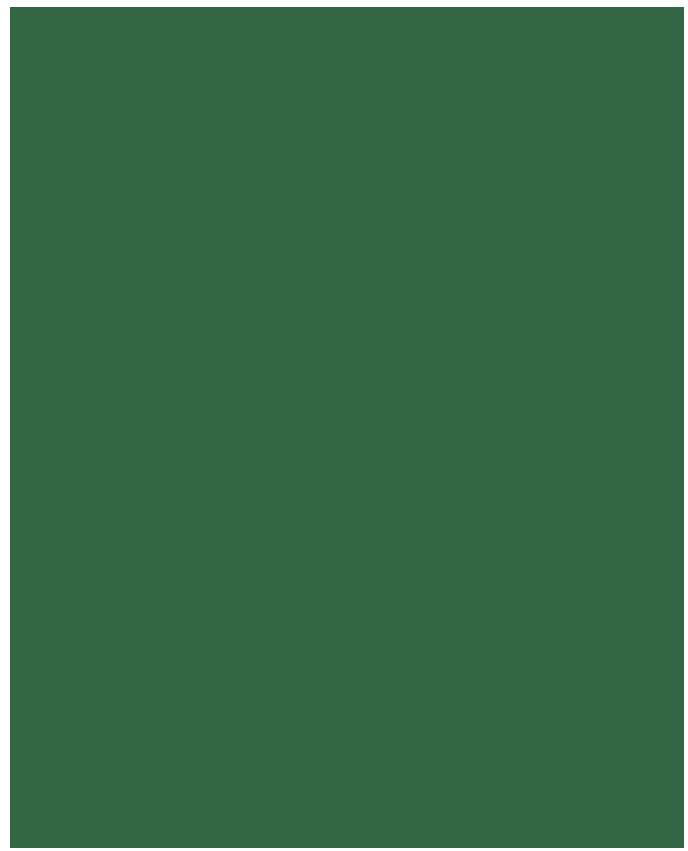
Finding the Maximum of a Function

For some function f of interest (e.g. the data likelihood) we would like to compute:

$$\hat{\theta} = \arg\max_{\theta} \; f(\theta)$$

When the function to optimize is continuously differentiable and concave, the maximum is found at the point where the gradient is zero.

$$\nabla_\theta f(\theta) = \begin{bmatrix} \partial f/\partial \theta_1 \\ \partial f/\partial \theta_2 \\ \vdots \\ \partial f/\partial \theta_h \end{bmatrix} = \mathbf{0}$$



Max Likelihood, Simple Gaussian Case

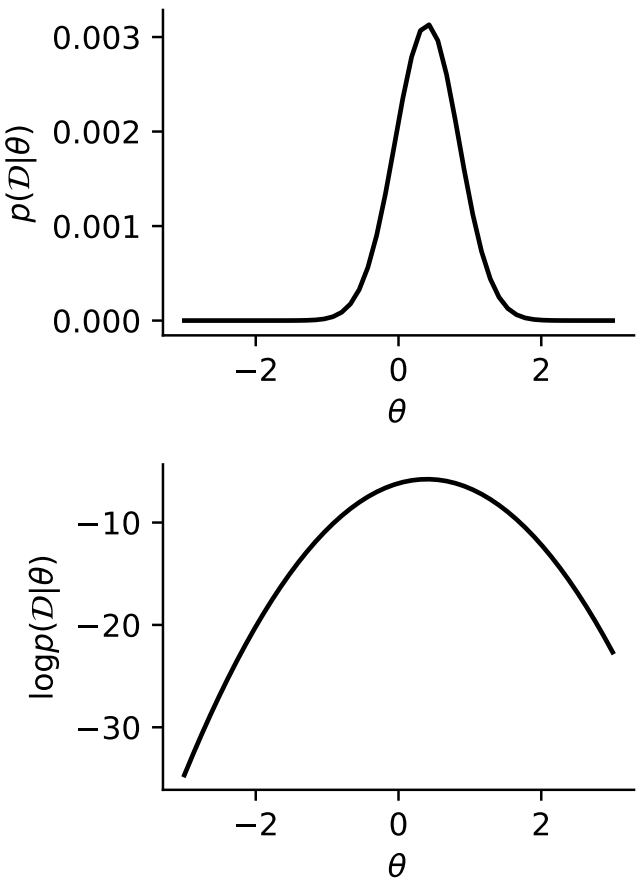
Observation: The function $p(\mathcal{D} \mid \theta)$ is not concave with θ.

Idea: Transform the function in a way that
(i) doesn't change its argmax and
(ii) makes it concave.

Applying the logarithm ensures the two properties above:

$$\log p(\mathcal{D} \mid \theta) = \log \prod_{k=1}^{N} \frac{1}{\sqrt{2\pi}} \exp\Big[-\frac{1}{2}(x_k - \theta)^2\Big]$$

$$= \sum_{k=1}^{N} \Big[-\frac{1}{2}\log(2\pi) - \frac{1}{2}(x_k - \theta)^2\Big]$$



Max Likelihood, Simple Gaussian Case

Having found the log-likelihood w.r.t. $\mathcal{D}$ to be

$$\log p(\mathcal{D} \,|\, \theta) = \sum_{k=1}^{N} \Big[-\frac{1}{2} \log(2\pi) - \frac{1}{2}(x_k - \theta)^2 \Big],$$

the best parameter $\hat{\theta}$ can then be found by solving $\nabla_\theta \log P(\mathcal{D} \,|\, \theta) = 0$.

Max Likelihood, Multivariate Case

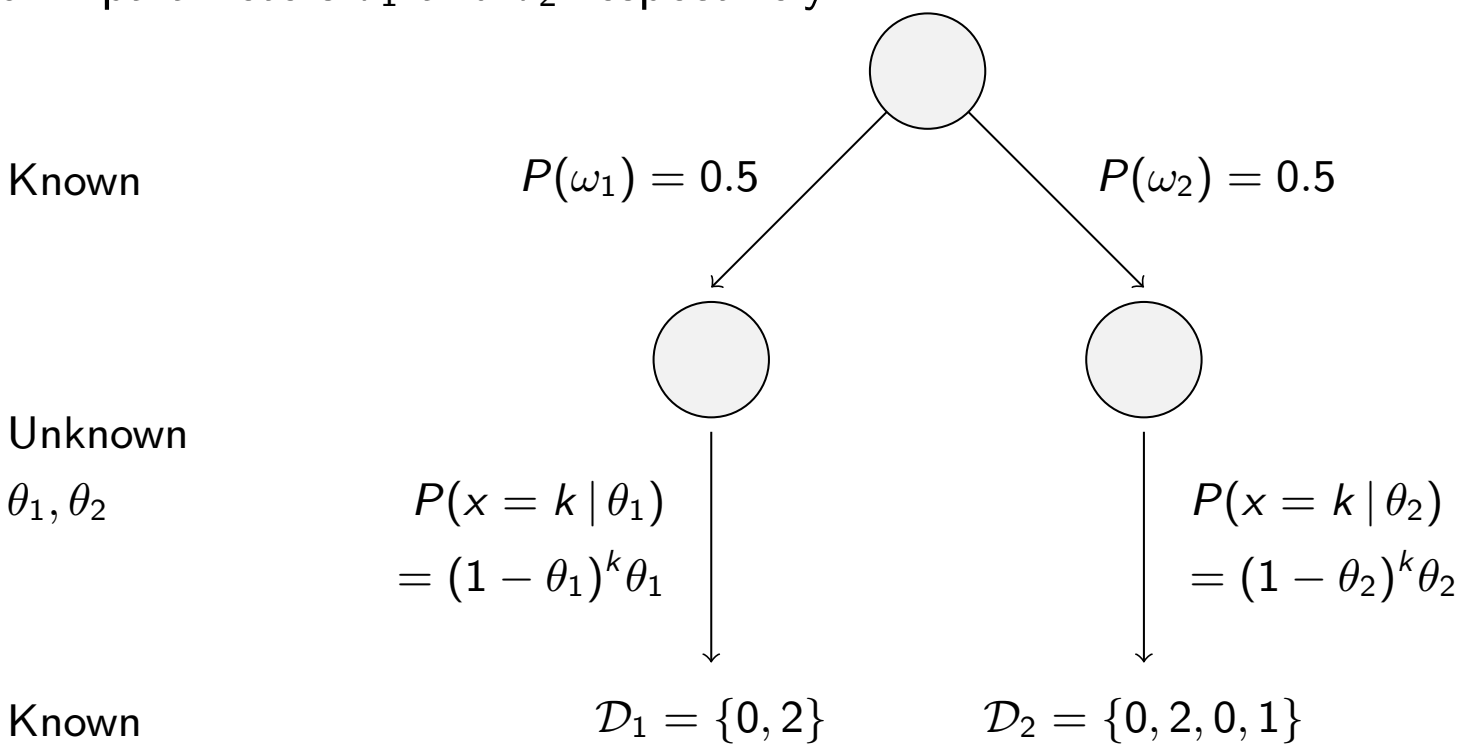
The log-likelihood of a *multivariate* Gaussian distribution w.r.t. $\mathcal{D}$ is given by

$$\log p(\mathcal{D} \mid \theta) = \sum_{k=1}^{N} -\frac{1}{2} \log \left[(2\pi)^d \det(\Sigma) \right] - \frac{1}{2} (\boldsymbol{x}_k - \boldsymbol{\mu})^\top \Sigma^{-1} (\boldsymbol{x}_k - \boldsymbol{\mu})$$

Question: Assuming Σ is fixed, what is the optimal parameter vector $\boldsymbol{\mu}$?

Building a Classifier with ML

Consider a labeled dataset containing two examples for the first class, and four examples for the second class. Points are in $\mathbb{N}_0$ and we assume they are generated from classes ω_1, ω_2 following geometric probability distributions of unknown parameters θ_1 and θ_2 respectively.



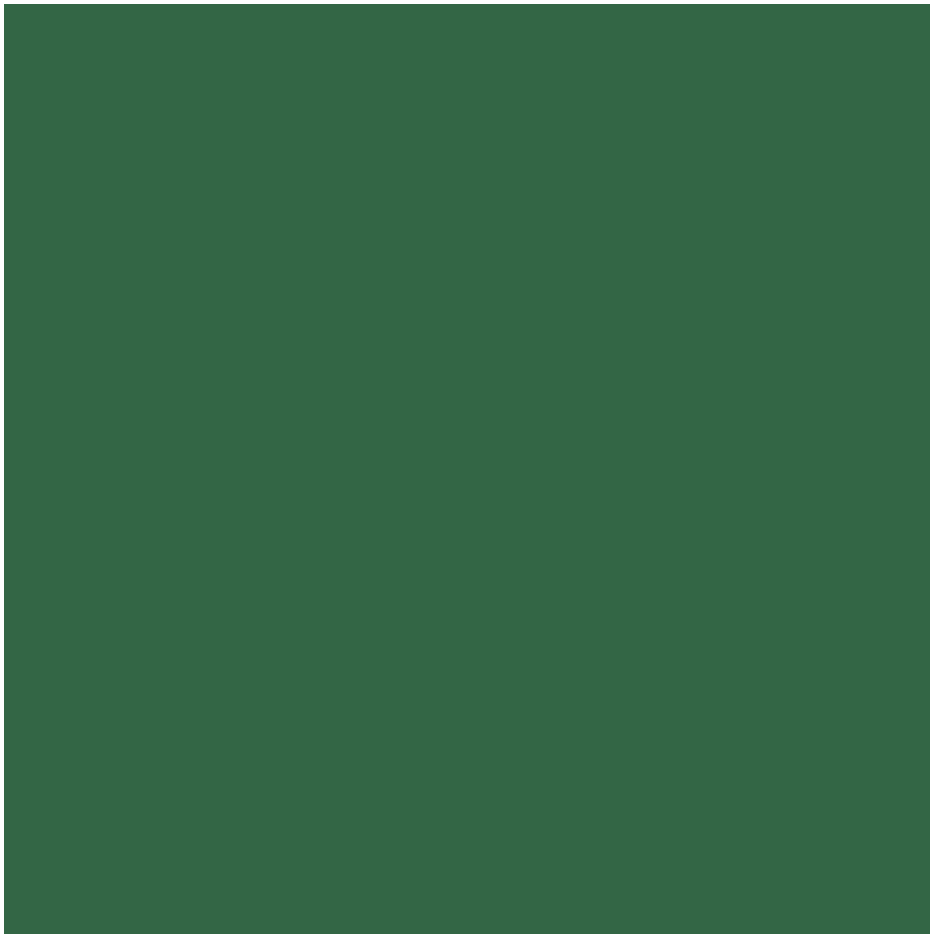
Building a Classifier with ML

Recall:

$$P(\omega_1) = 0.5$$
$$P(\omega_2) = 0.5$$

$$P(x = k \mid \theta_1) = (1 - \theta_1)^k \theta_1$$
$$P(x = k \mid \theta_2) = (1 - \theta_2)^k \theta_2$$

$$\mathcal{D}_1 = \{0, 2\}$$
$$\mathcal{D}_2 = \{0, 2, 0, 1\}$$



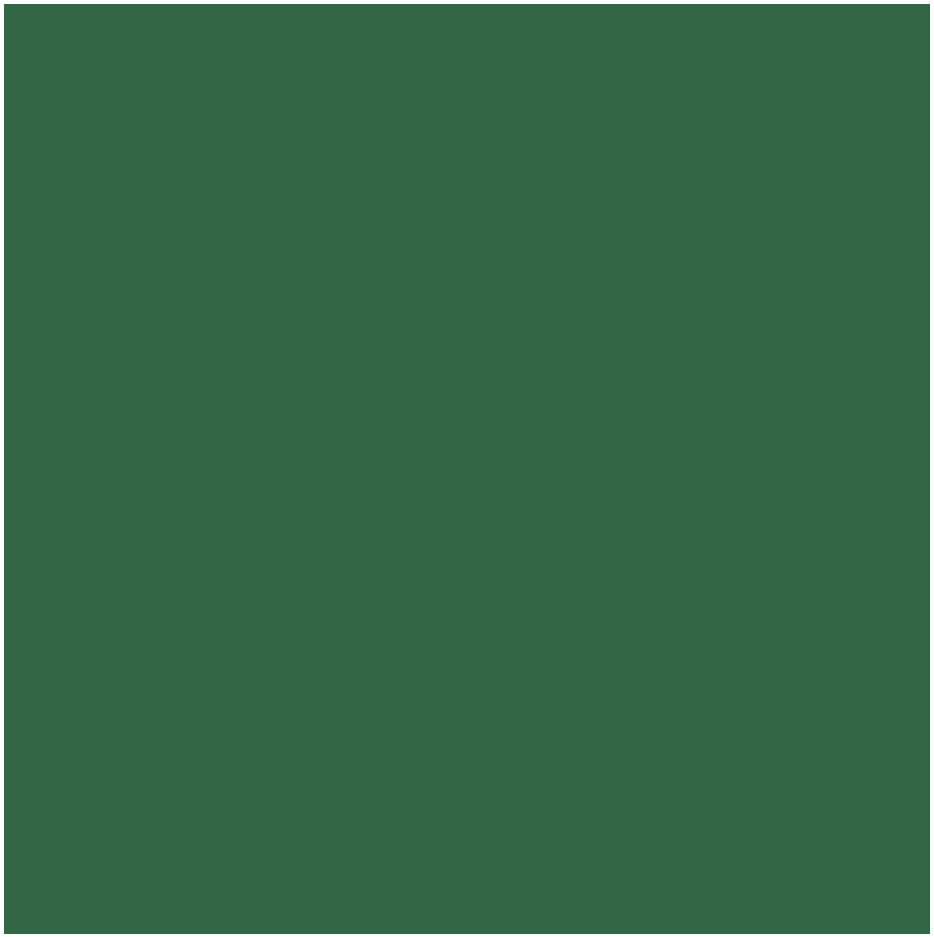
Building a Classifier with ML

Recall:

$$P(\omega_1) = 0.5$$
$$P(\omega_2) = 0.5$$

$$P(x = k \mid \theta_1) = (1 - \theta_1)^k \theta_1$$
$$P(x = k \mid \theta_2) = (1 - \theta_2)^k \theta_2$$

$$\mathcal{D}_1 = \{0, 2\}$$
$$\mathcal{D}_2 = \{0, 2, 0, 1\}$$



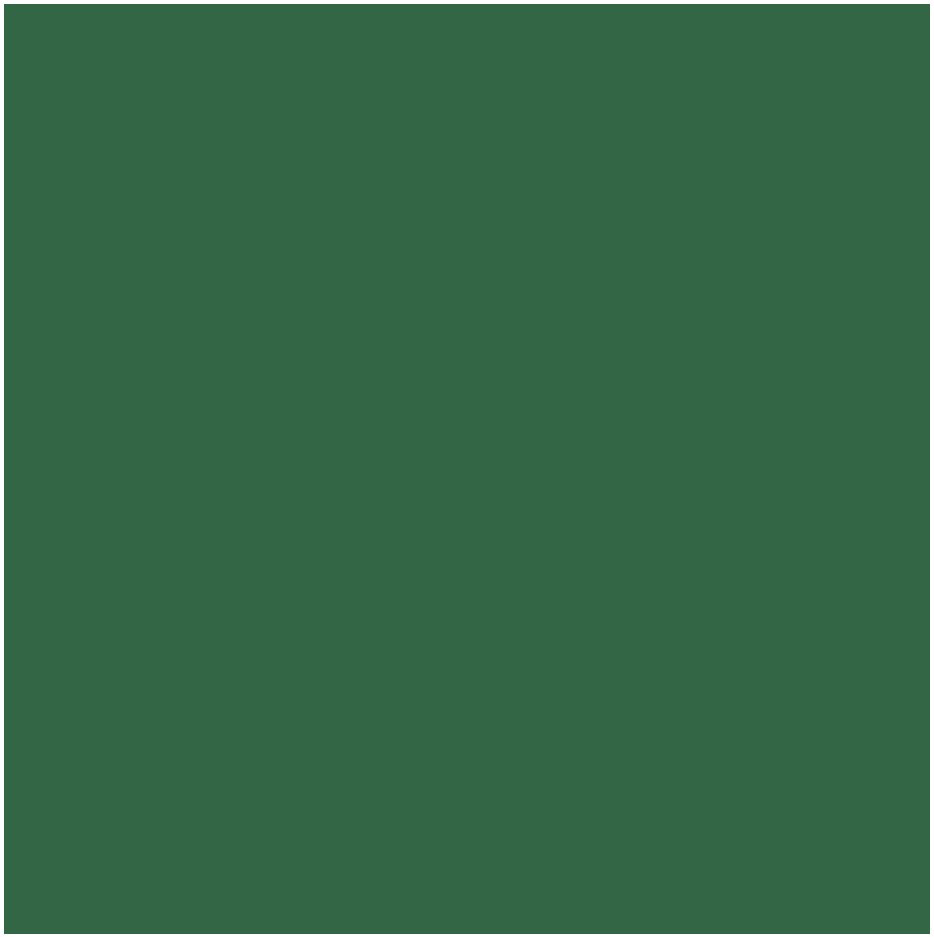
Building a Classifier with ML

Recall:

$$P(\omega_1) = 0.5$$
$$P(\omega_2) = 0.5$$

$$P(x = k \mid \theta_1) = (1 - \theta_1)^k \theta_1$$
$$P(x = k \mid \theta_2) = (1 - \theta_2)^k \theta_2$$

$$\mathcal{D}_1 = \{0, 2\}$$
$$\mathcal{D}_2 = \{0, 2, 0, 1\}$$



Bayes Parameter Estimation

Assuming some dataset $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$. We would like to model this dataset using some probability function $p(\mathcal{D}|\theta)$ with θ some unknown parameter that needs to be learned.

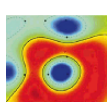
ML parameter estimation: Chose the parameter θ that maximises the data likelihood:

$$\hat{\theta} = \arg\max_{\theta} p(\mathcal{D}|\theta)$$

Bayes parameter estimation: Instead of learning a fixed estimate, build a posterior distribution over this parameter using the Bayes theorem:

$$\begin{aligned} p(\theta|\mathcal{D}) &= \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \\ &= \frac{p(\mathcal{D}|\theta)p(\theta)}{\int_\theta p(\mathcal{D}|\theta)p(\theta)d\theta} \end{aligned}$$

This approach requires to define a *prior* distribution $p(\theta)$ which models our initial belief about the parameter θ before observing the data.



From ML to Bayes Classifiers

ML classifier: Class posteriors are given by:

$$P(\omega_i \,|\, \boldsymbol{x}) = \frac{p(\boldsymbol{x} \,|\, \omega_i; \hat{\theta}_i)\, P(\omega_i)}{\sum_i p(\boldsymbol{x} \,|\, \omega_i; \hat{\theta}_i)\, P(\omega_i)}$$

where $\hat{\theta}_i$ is the maximum likelihood parameter for the distribution of class ω_i.

Bayes classifier: Class posteriors are computed by bypassing the intermediate computation of the parameters $\hat{\theta}_i$, and instead conditioning directly on the data:

$$\begin{aligned} P(\omega_i \,|\, \boldsymbol{x}, \mathcal{D}) &= \frac{p(\boldsymbol{x} \,|\, \omega_i, \mathcal{D})\, P(\omega_i \,|\, \mathcal{D})}{\sum_i p(\boldsymbol{x} \,|\, \omega_i, \mathcal{D})\, P(\omega_i \,|\, \mathcal{D})} \\ &= \frac{p(\boldsymbol{x} \,|\, \omega_i, \mathcal{D}_i)\, P(\omega_i)}{\sum_i p(\boldsymbol{x} \,|\, \omega_i, \mathcal{D}_i)\, P(\omega_i)} \end{aligned}$$

Bayes Classifiers (cont.)

The terms of the class posterior:

$$P(\omega_i \,|\, \boldsymbol{x}, \mathcal{D}) = \frac{p(\boldsymbol{x} \,|\, \omega_i, \mathcal{D}_i)\, P(\omega_i)}{\sum_i p(\boldsymbol{x} \,|\, \omega_i, \mathcal{D}_i)\, P(\omega_i)}$$

can be expressed with model parameters as:

$$\begin{aligned} p(\boldsymbol{x} \,|\, \omega_i, \mathcal{D}_i) &= \int p(\boldsymbol{x} \,|\, \theta_i, \omega_i, \mathcal{D}_i)\, p(\theta_i \,|\, \omega_i, \mathcal{D}_i) d\theta_i \\ &= \int p(\boldsymbol{x} \,|\, \theta_i)\, p(\theta_i \,|\, \mathcal{D}_i) d\theta_i \end{aligned}$$

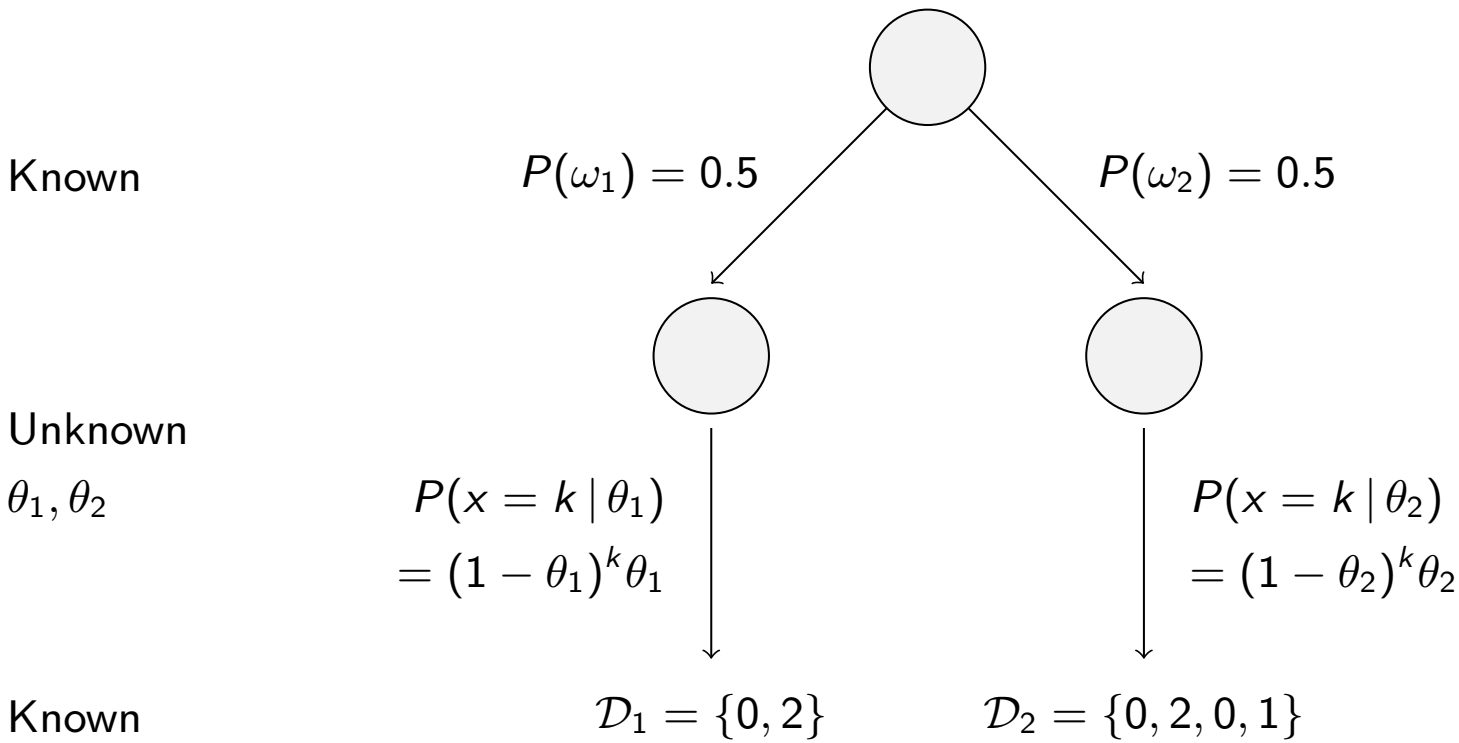
and

$$p(\theta_i \,|\, \mathcal{D}_i) = \frac{p(\mathcal{D}_i \,|\, \theta_i)\, p(\theta_i)}{\int p(\mathcal{D}_i \,|\, \theta_i)\, p(\theta_i) d\theta_i}.$$

are our Bayes parameter estimates.

Building a Classifier with Bayes

Recall: we consider the following data generating process



Building a Classifier with Bayes

Recall:

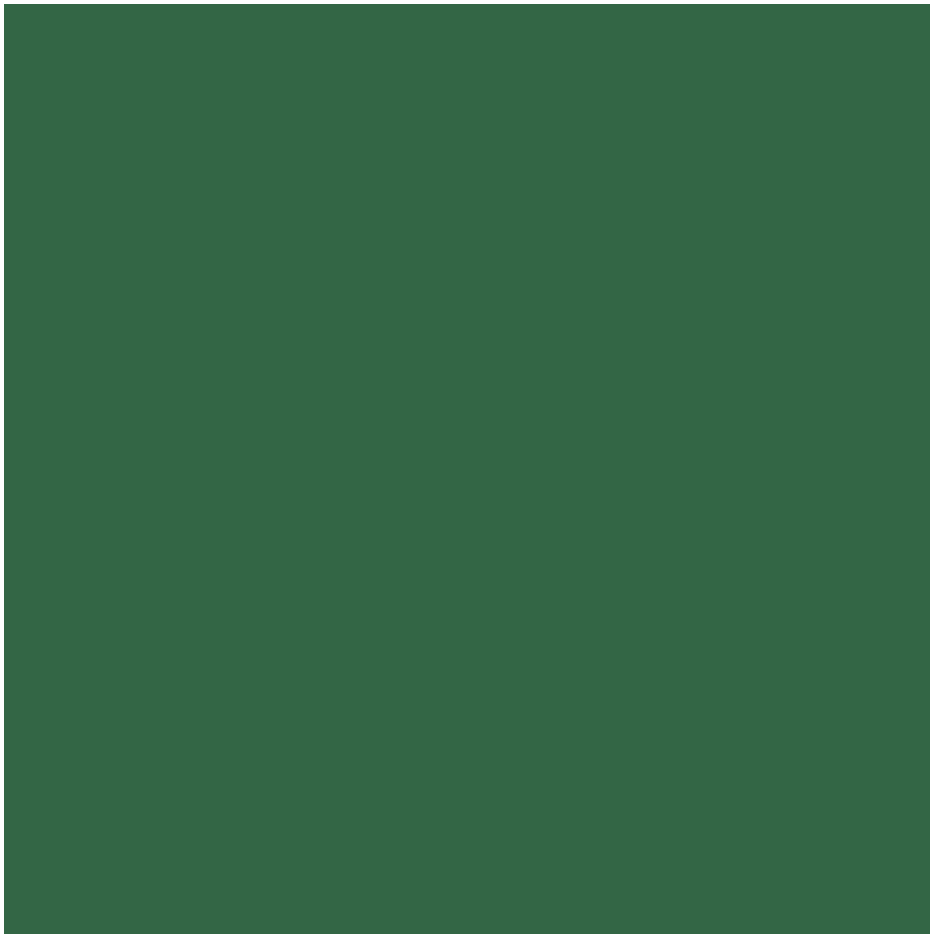
$$P(\omega_1) = 0.5$$
$$P(\omega_2) = 0.5$$

$$P(x = k \mid \theta_1) = (1 - \theta_1)^k \theta_1$$
$$P(x = k \mid \theta_2) = (1 - \theta_2)^k \theta_2$$

$$\mathcal{D}_1 = \{0, 2\}$$
$$\mathcal{D}_2 = \{0, 2, 0, 1\}$$

... and we further set:

$$p(\theta_1) \sim \mathcal{U}(0, 1)$$
$$p(\theta_2) \sim \mathcal{U}(0, 1)$$



Building a Classifier with Bayes

Recall:

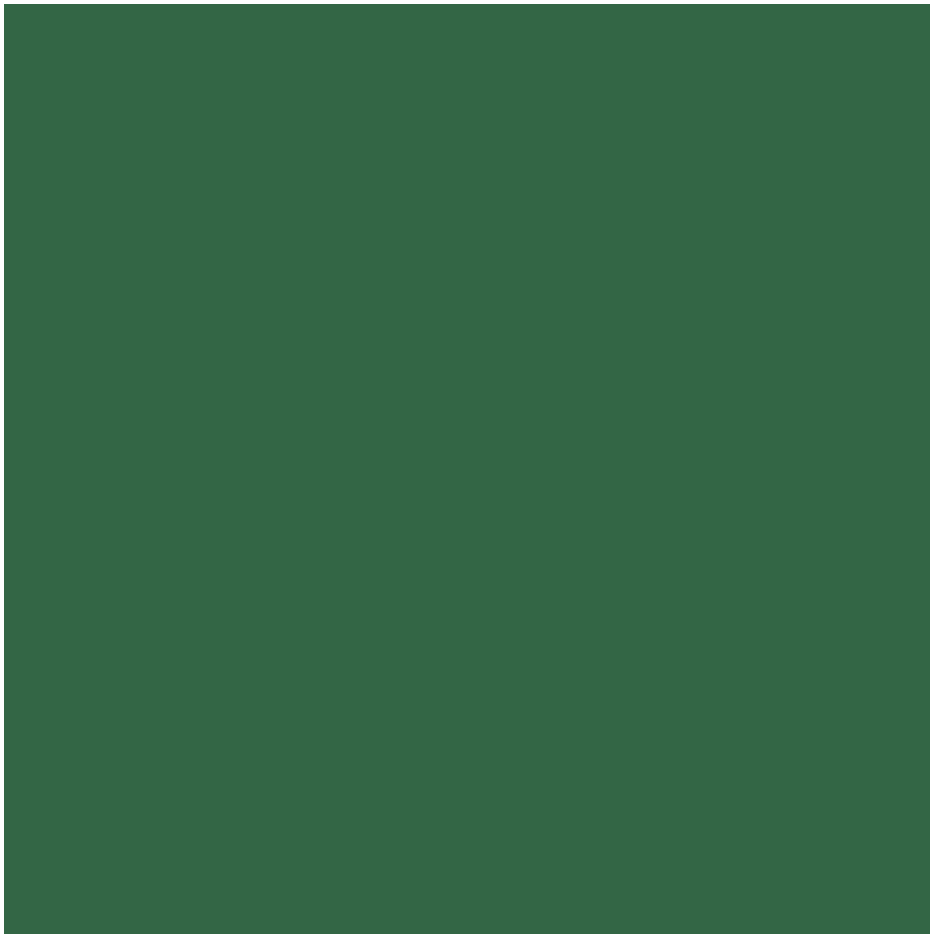
$$P(\omega_1) = 0.5$$
$$P(\omega_2) = 0.5$$

$$P(x = k \mid \theta_1) = (1 - \theta_1)^k \theta_1$$
$$P(x = k \mid \theta_2) = (1 - \theta_2)^k \theta_2$$

$$\mathcal{D}_1 = \{0, 2\}$$
$$\mathcal{D}_2 = \{0, 2, 0, 1\}$$

... and we further set:

$$p(\theta_1) \sim \mathcal{U}(0, 1)$$
$$p(\theta_2) \sim \mathcal{U}(0, 1)$$



Building a Classifier with Bayes

Recall:

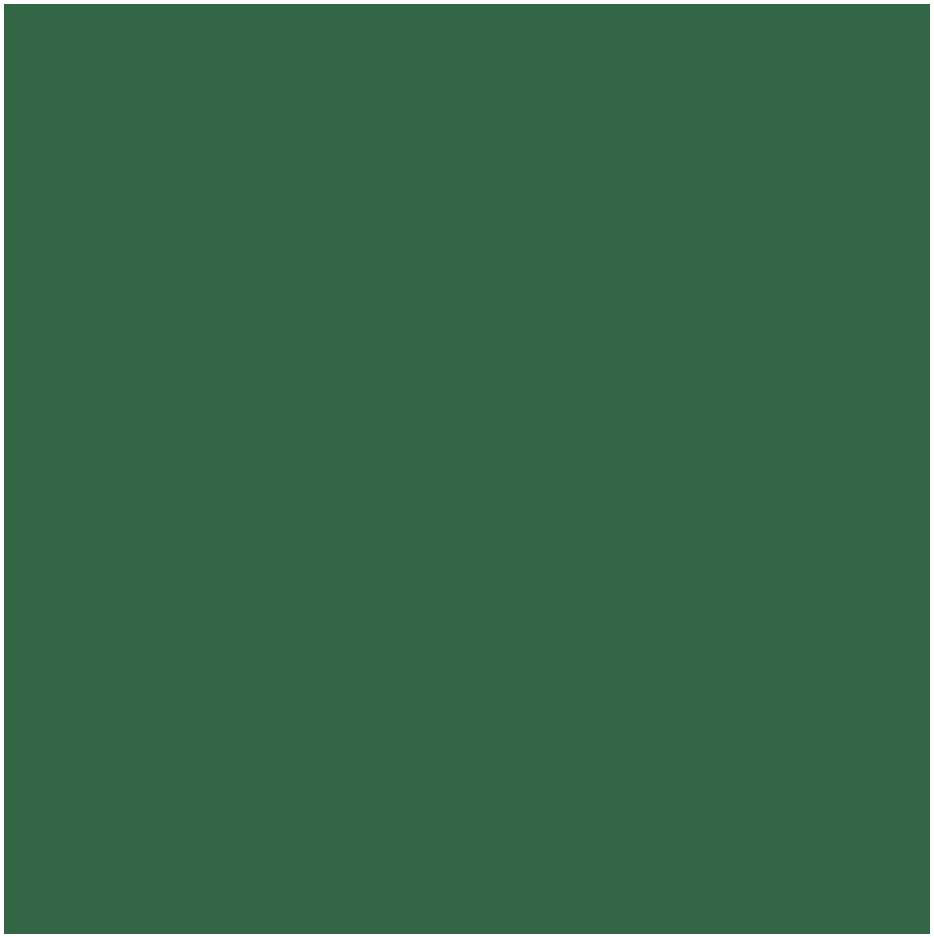
$$P(\omega_1) = 0.5$$
$$P(\omega_2) = 0.5$$

$$P(x = k \mid \theta_1) = (1 - \theta_1)^k \theta_1$$
$$P(x = k \mid \theta_2) = (1 - \theta_2)^k \theta_2$$

$$\mathcal{D}_1 = \{0, 2\}$$
$$\mathcal{D}_2 = \{0, 2, 0, 1\}$$

... and we further set:

$$p(\theta_1) \sim \mathcal{U}(0, 1)$$
$$p(\theta_2) \sim \mathcal{U}(0, 1)$$



Building a Classifier with Bayes

Recall:

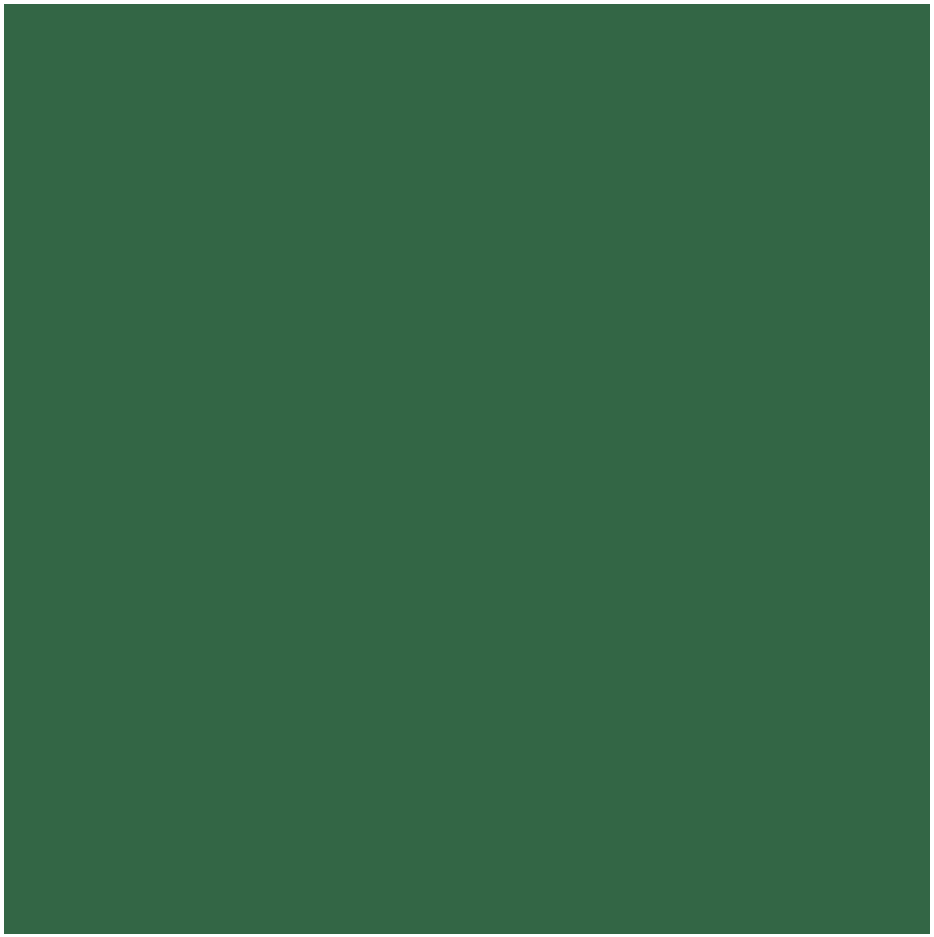
$$P(\omega_1) = 0.5$$
$$P(\omega_2) = 0.5$$

$$P(x = k \mid \theta_1) = (1 - \theta_1)^k \theta_1$$
$$P(x = k \mid \theta_2) = (1 - \theta_2)^k \theta_2$$

$$\mathcal{D}_1 = \{0, 2\}$$
$$\mathcal{D}_2 = \{0, 2, 0, 1\}$$

... and we further set:

$$p(\theta_1) \sim \mathcal{U}(0, 1)$$
$$p(\theta_2) \sim \mathcal{U}(0, 1)$$



ML vs. Bayes Classifiers

Observations:

- ML and Bayes classifiers do not always produce the same decisions
- Bayes classifiers are influenced by the prior distribution and are consequently less sensitive to the data.
- Bayes classifiers will tend to favor the outcome that is supported by a larger amount of data.

ML vs. Bayes: Gaussian Case

Consider the simple data density $p(x \mid \mu) = \mathcal{N}(\mu, \sigma^2)$ with unknown parameter μ. We would like to compare the ML and Bayes approaches to estimate the parameter μ from some dataset $\mathcal{D} = \{x_1, \ldots, x_n\}$.

ML approach:

- The maximum likelihood estimate is given by $\hat{\mu} = \frac{1}{n} \sum_{k=1}^{n} x_k$, i.e. the empirical mean (cf. previous slides).

Bayes approach:

- Assuming some prior distribution $p(\mu) = \mathcal{N}(\mu_0, \sigma_0^2)$, the posterior distribution can be computed as:

$$p(\mu \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \mu) p(\mu)}{p(\mathcal{D})} = \alpha \prod_{k=1}^{n} p(x_k|\mu) p(\mu)$$

where α is a normalizing factor.

ML vs. Bayes: Gaussian Case

Bayes approach (continued):

- The posterior distribution can be expanded as:

$$\begin{aligned} p(\mu|\mathcal{D}) &= \alpha \prod_{k=1}^{n} \overbrace{\frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x_k-\mu}{\sigma}\right)^2\right]}^{p(x_k|\mu)} \overbrace{\frac{1}{\sqrt{2\pi}\sigma_0} \exp\left[-\frac{1}{2}\left(\frac{\mu-\mu_0}{\sigma_0}\right)^2\right]}^{p(\mu)} \\ &= \alpha' \exp\left[-\frac{1}{2}\left(\sum_{k=1}^{n}\left(\frac{\mu-x_k}{\sigma}\right)^2 + \left(\frac{\mu-\mu_0}{\sigma_0}\right)^2\right)\right] \\ &= \alpha'' \exp\left[-\frac{1}{2}\left[\left(\frac{n}{\sigma^2}+\frac{1}{\sigma_0^2}\right)\mu^2 - 2\left(\frac{1}{\sigma^2}\sum_{k=1}^{n} x_k + \frac{\mu_0}{\sigma_0^2}\right)\mu\right]\right], \end{aligned}$$

  which corresponds to a new Gaussian distribution $\mathcal{N}(\mu_n, \sigma_n^2)$ with parameters

$$\mu_n = \frac{\sigma_n^2}{\sigma^2/n}\hat{\mu} + \frac{\sigma_n^2}{\sigma_0^2}\mu_0 \quad \text{and} \quad \sigma_n^2 = \left(\frac{1}{\sigma^2/n} + \frac{1}{\sigma_0^2}\right)^{-1}$$

- We observe that (1) the mean estimate μ_n is now pulled towards the prior if too little data is available, and (2) the mean estimate comes with a variance term σ_n^2 that can be useful for confidence estimation.

Summary

- In practice, parameters of the class-conditioned distributions are not known and they must be inferred from some finite dataset.
- Two main approaches for learning these parameters: (1) maximum likelihood estimation and (2) Bayesian inference.
- Bayesian inference is often more difficult than maximum likelihood estimation (both analytically and computationally), because it requires integration.
- Bayesian inference readily incorporates interesting functionalities (inclusion of priors, construction of confidence estimates).