8. Model Inference and Averaging

오영민

Introduction

- The Bootstrap and Maximum Likelihood Methods
- Bayesian Methods
- MCMC for sampling the Posterior
- The EM Algorithm
- Bagging
- Model Averaging and Stacking

Maximum Likelihood Inference

- For most of this chapters, the fitting of models has been achieved by minimizing a sum of squares for regression, or by minimizing cross-entropy for classification

- In fact, both of these minimizations are **instances of the maximum likelihood approach** to fitting.

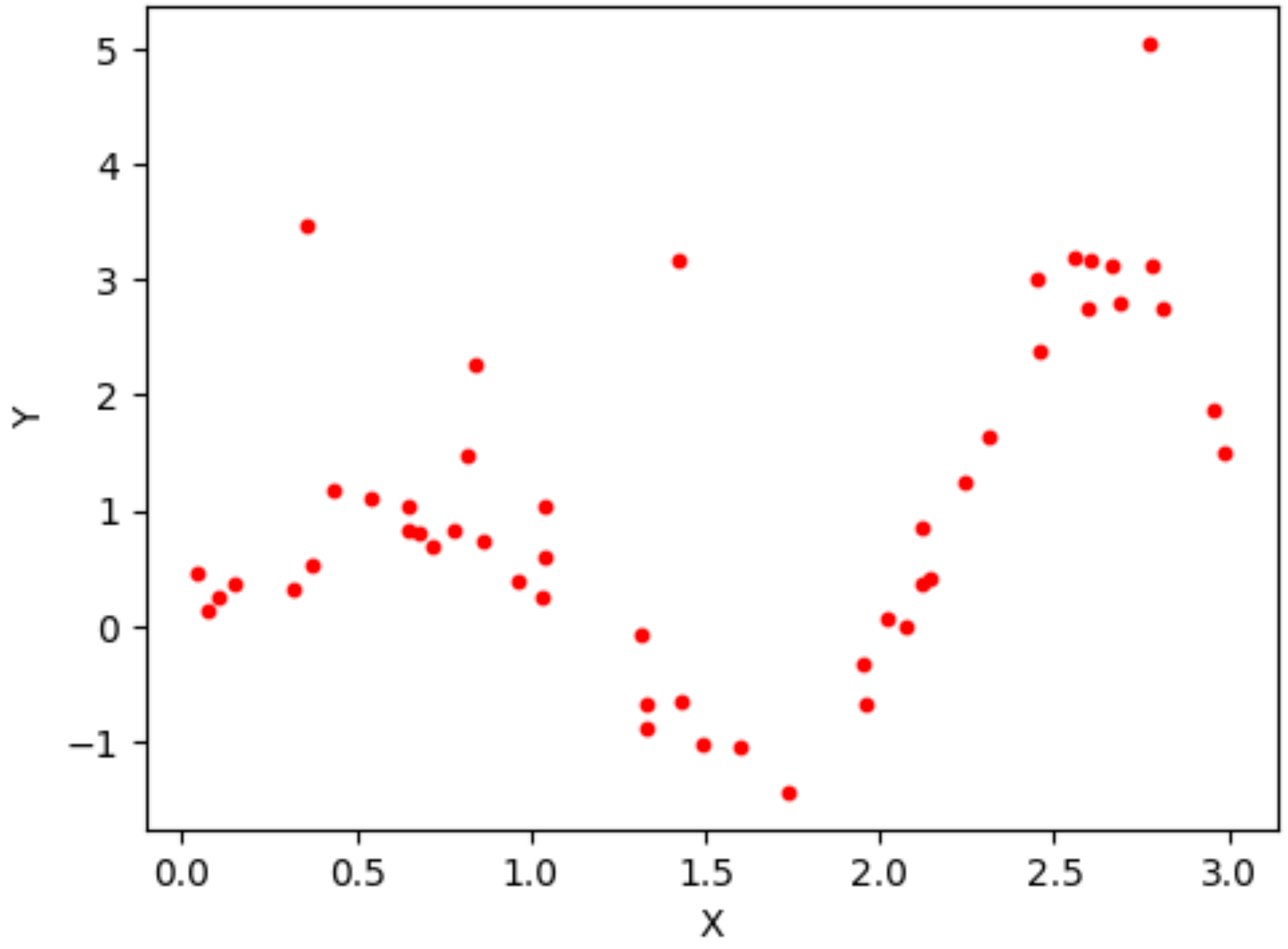
- In this chapter, we discuss a general exposition of the maximum likelihood approach, as well as the Bayesian method for inference.

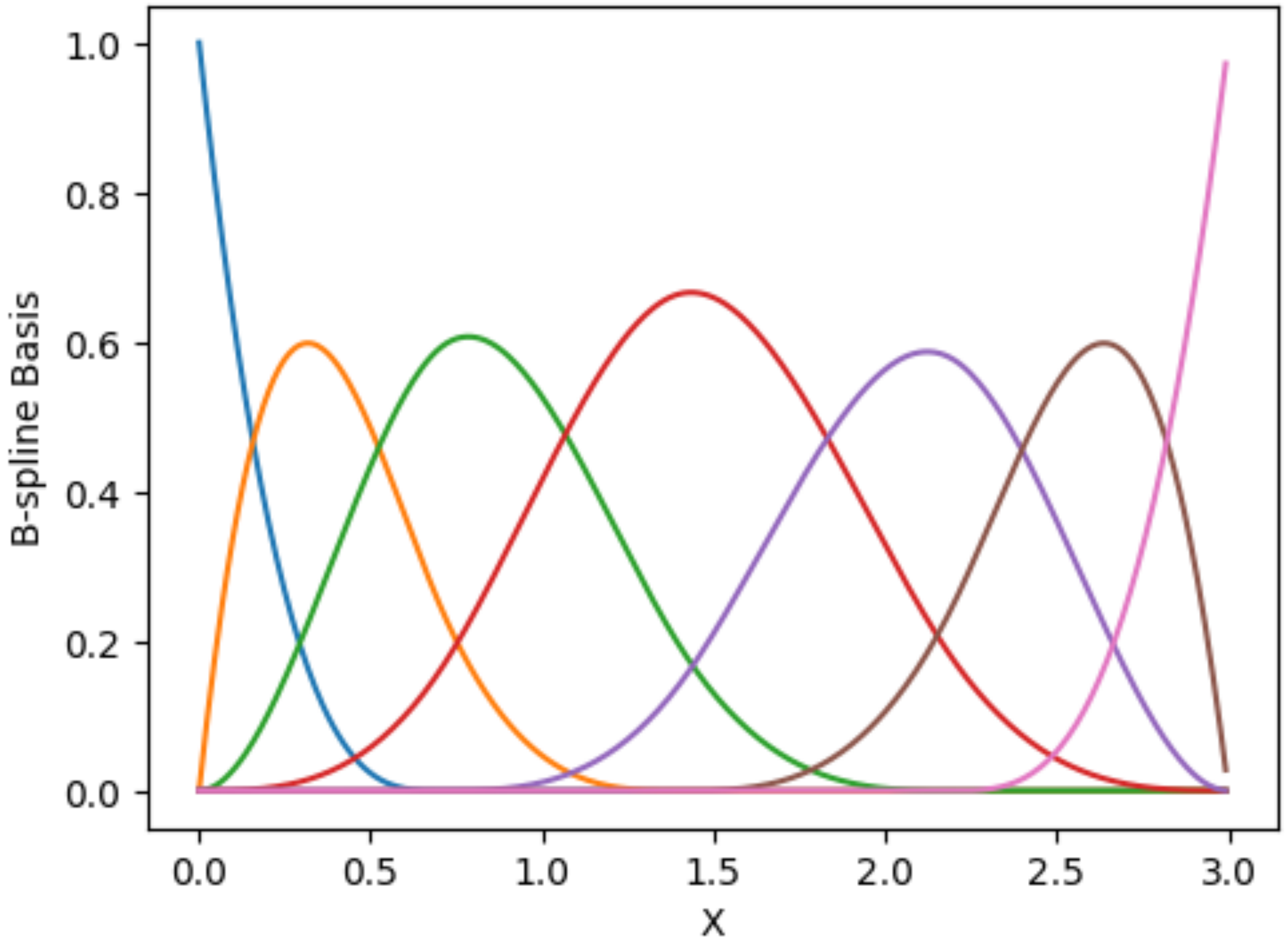
Maximum Likelihood Inference

- Let our observations $Z = \{z_i\}_{i=1}^N$ where $z_i = (x_i, y_i) \sim g_\theta(z), \forall i$.
- **(The information matrix)** $\mathbf{I}(\theta) = -\sum_{i=1}^{N} \frac{\partial^2 l(\theta; z_i)}{\partial\theta\partial\theta^T}$
- **(The Fisher information)** $\mathbf{i}(\theta) = \mathbb{E}_\theta\left[\mathbf{I}(\theta)\right]$
- For iid z_i and under general conditions, $(\hat{\theta} - \theta_0) \xrightarrow{D} \mathcal{N}\left(0, \mathbf{I}^{-1}(\theta_0)\right)$ as $N \to \infty$ where θ_0 is true value of θ, $\hat{\theta}$ is maximum likelihood estimator.
- **Approximately,** $(1-\alpha)\,\%$ **of CI for** θ_j **:** $\hat{\theta}_j \pm z_{\alpha/2}\sqrt{\mathbf{i}^{-1}(\hat{\theta})_{jj}}$ **or** $\hat{\theta}_j \pm z_{\alpha/2}\sqrt{\mathbf{I}^{-1}(\hat{\theta})_{jj}}$

Maximum Likelihood Inference

- **(A Smoothing Example)** Let $x_i \in \mathbb{R}$, $N = 50$ with 3-knots placed at the quartiles, by B-spline basis functions: $\mu(x) = \sum_{j=1}^{7} \beta_j h_j(x)$. Assume that $Y = \mu(X) + \epsilon$,$\epsilon \sim \mathcal{N}(0,\sigma^2)$



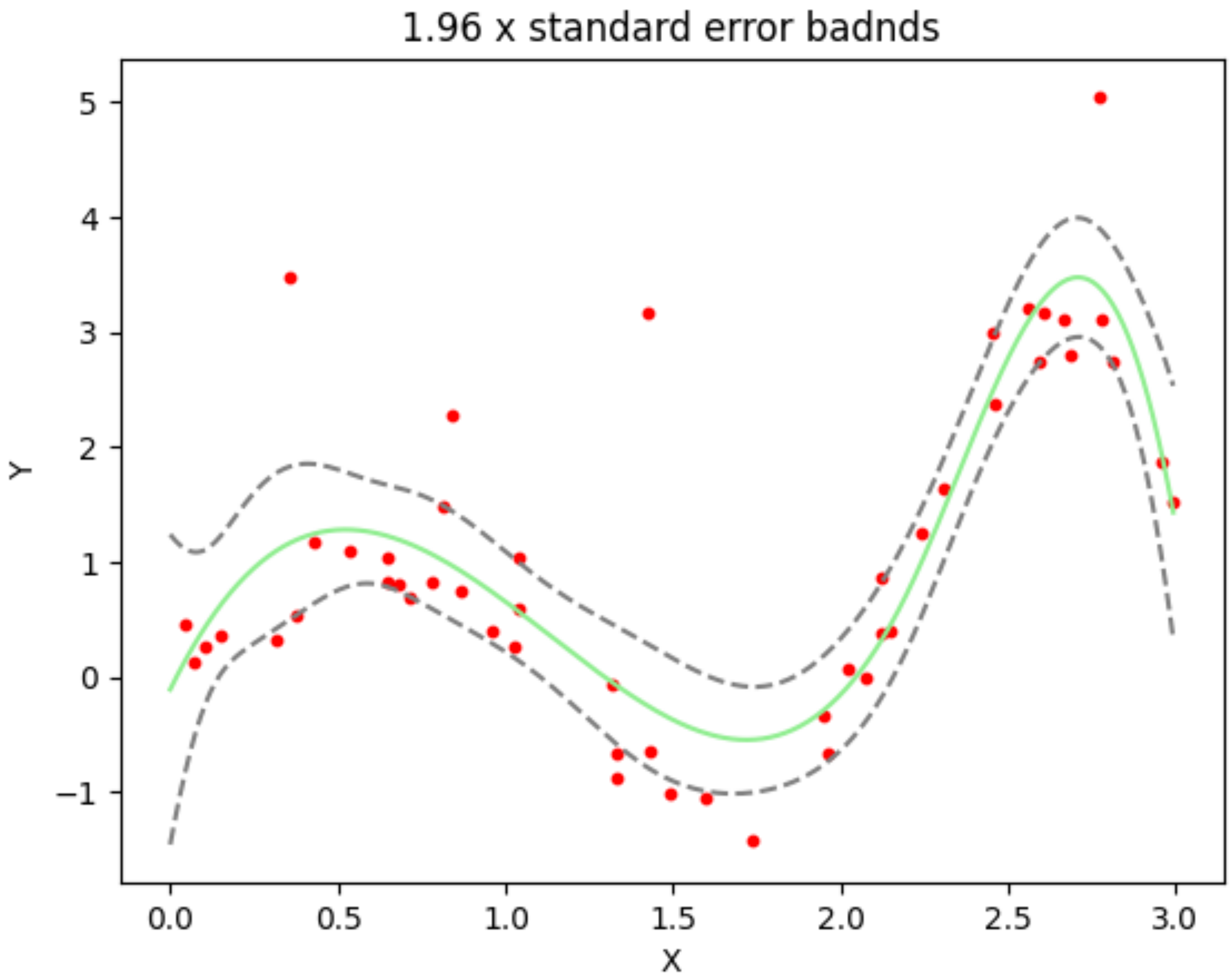


Maximum Likelihood Inference

- Then $\theta = (\beta, \sigma^2)$, $l(\theta) = -\frac{N}{2} log \sigma^2 2\pi - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - h(x_i)^T\beta)^2$
- By setting $\frac{\partial l}{\partial \beta} = 0$ and $\frac{\partial l(\hat{\beta})}{\partial \sigma^2} = 0$, giving $\hat{\beta} = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}\mathbf{y}$, $\hat{\sigma}^2 = \frac{1}{N}\sum\left(y_i - \hat{\mu}(x_i)\right)^2$
- The information matrix is $\mathbf{I}(\beta) = \left(\mathbf{H}^T\mathbf{H}\right)/\sigma^2$
- $(1-\alpha)\,\%$ for $\mu(x) = h(x)^T\beta$: $\hat{\mu}(x) \pm z_{\alpha/2}\sqrt{h(x)^T\left(\mathbf{H}^T\mathbf{H}\right)^{-1}h(x)\hat{\sigma}^2}$

Note. Least square estimate is $\hat{\beta} = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}\mathbf{y}$ and $\hat{Var}(\hat{\beta}) = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\hat{\sigma}^2$. Thus, the estimates of information matrix agrees with the least squares estimate.

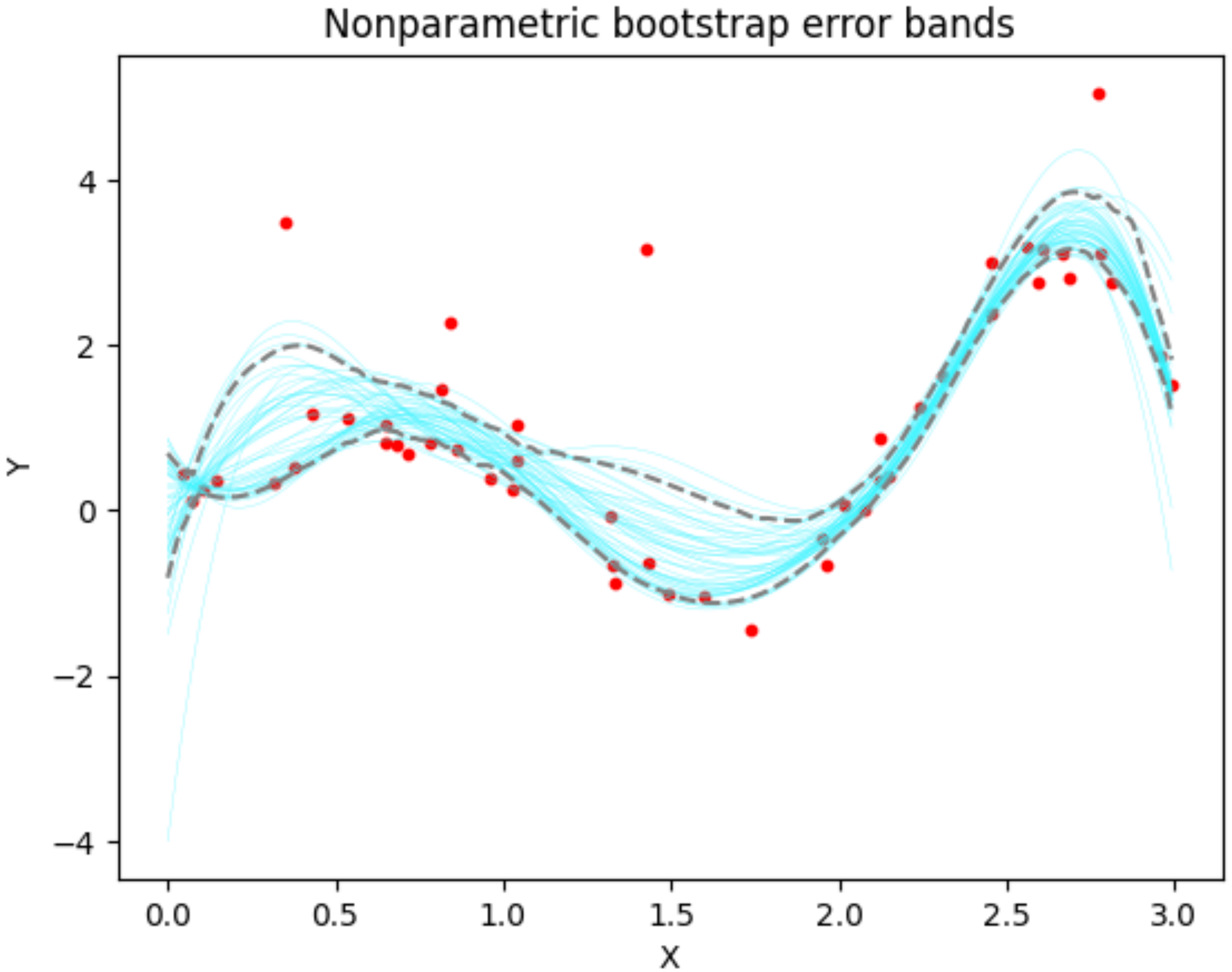
Maximum Likelihood Inference



Non-parametric Bootstrap

1. Draw B datasets each of size $N = 50$ with replacement: $\mathbf{Z}^{b*} = \left\{z_i^{b*}\right\}_{i=1}^{50}$, $b = 1,\ldots,B$

2. To each dataset, we fit $\hat{\mu}^*(x)$ and form a $(1-\alpha)\,\%$ point wise confidence band from the percentiles at each x.



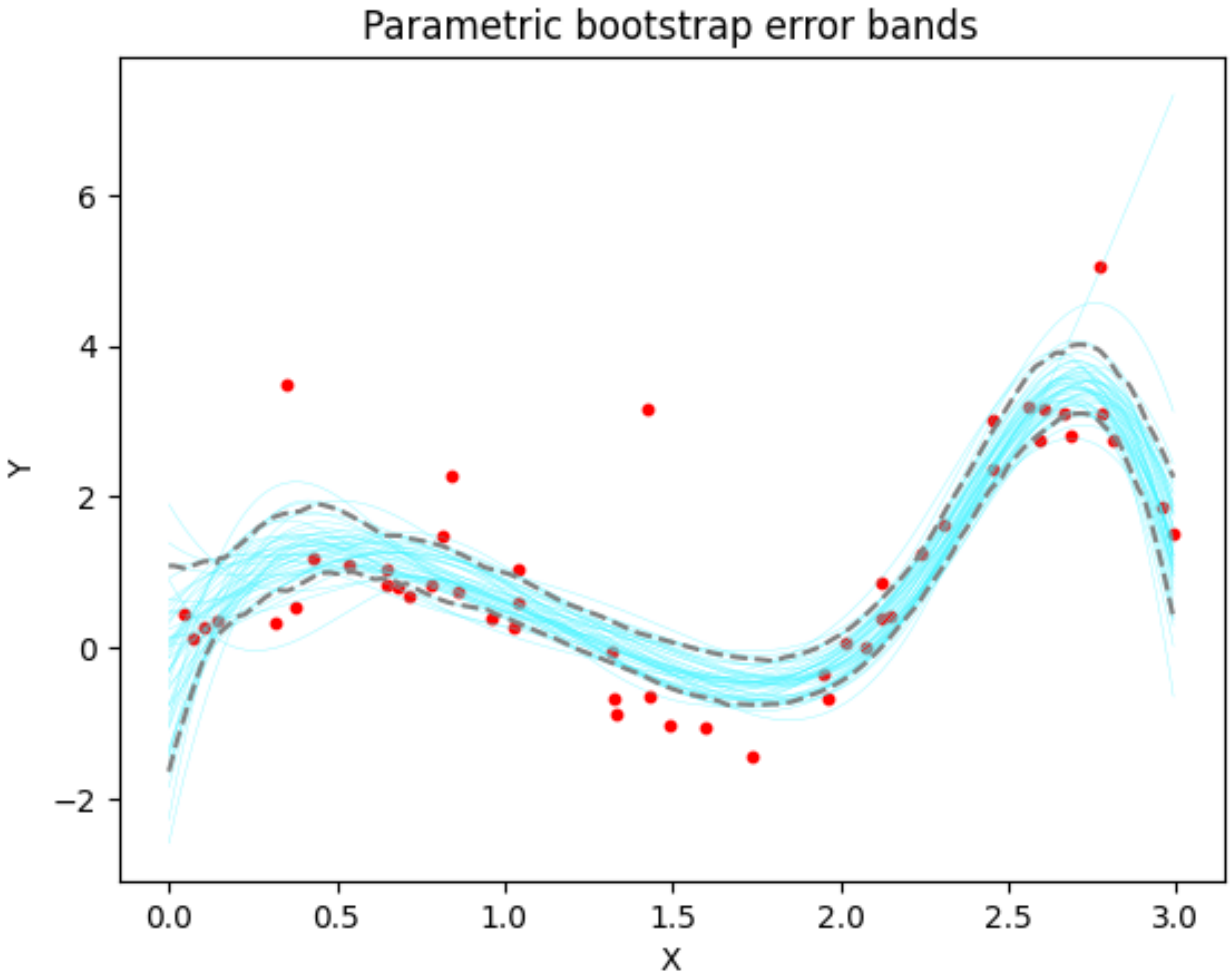
Parametric Bootstrap

1. Simulate new responses by $y_i^* = \hat{\mu}(x_i) + \epsilon_i^*$, $\epsilon_i^* \sim \mathcal{N}(0,\hat{\sigma}^2)$ $i = 1,...,50$ replicates B-times.

2. To each dataset, $\left\{(x_i, y_i^*)\right\}_{i=1}^{50}$ we fit $\hat{\mu}^*(x)$ and form a $(1-\alpha)\,\%$ point wise confidence band from the percentiles at each x.

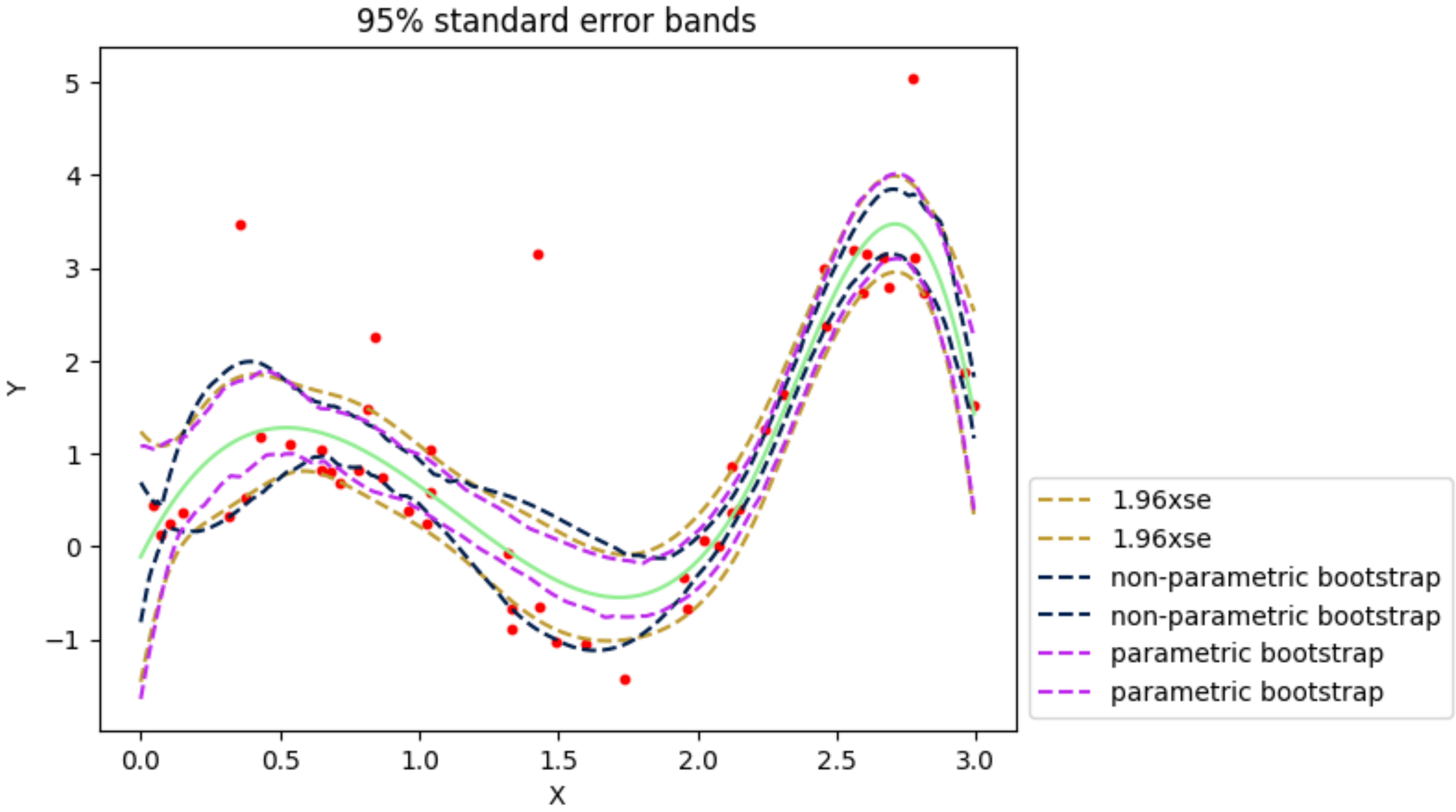
- Then a function estimated from $\mathbf{y}^*$: $\hat{\mu}^*(x) = h(x)^T\hat{\beta}^* = h(x)^T(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{y}^*$. Since $\mathbb{E}(\mathbf{y}^*) = \hat{\mu}(\mathbf{X}) = \mathbf{H}\hat{\beta}$, $\hat{\mu}^*(x) \sim \mathcal{N}(\hat{\mu}(x), h(x)^T(\mathbf{H}^T\mathbf{H})^{-1}h(x)\hat{\sigma}^2)$ which is the same as maximum likelihood.

Parametric Bootstrap



Parametric Bootstrap

- In essence the bootstrap is a computer implementation of non-parametric or parametric maximum likelihood.



Bayesian Methods

- We specify a sampling model $Pr(\mathbf{Z} | \theta)$ and a prior $Pr(\theta)$ **reflecting our knowledge about θ before we see the data**. Then compute $Pr(\theta | \mathbf{Z}) \propto Pr(\mathbf{Z} | \theta) \cdot Pr(\theta)$ which represents our updated knowledge about θ after we see the data.

- The predictive distribution: $Pr(z^{new} | \mathbf{Z}) = \int Pr(z^{new} | \theta) \cdot Pr(\theta | \mathbf{Z}) d\theta$ which **accounts for the uncertainty in estimating θ.** (In the maximum likelihood approach would use $Pr(z^{new} | \hat{\theta})$.)

- In smoothing example, assume that σ^2 is known, and x_i are fixed, so that randomness in the data comes solely from y.

- Distribution of prior $\mu(x)$: Instead provide a prior for $\beta \sim \mathcal{N}(0, \tau\Sigma)$. ($\because$ Distributions on functions are fairly complex.) Then $\mu(x) \sim \mathcal{N}(0, \tau \mathbf{H} \Sigma \mathbf{H}^T)$.

Bayesian Methods

- **The posterior distribution for β:** Since $y_i = h(x_i)^T\beta + \epsilon \sim \mathcal{N}(h(x_i)^T\beta, \sigma^2)$, $\mathbf{y}\,|\,\beta \sim \mathcal{N}(\mathbf{H}\beta, \sigma^2\mathbf{I})$

$$Pr(\beta\,|\,\mathbf{Z}) \propto Pr(\beta)Pr(\mathbf{Z}\,|\,\beta) \propto exp\left[-\frac{1}{2\tau}\beta^T\Sigma^{-1}\beta - \frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{H}\beta)^T(\mathbf{y}-\mathbf{H}\beta)\right]$$

$$\propto exp\left[\beta^T\underline{(\mathbf{H}^T\mathbf{H} + \frac{\sigma^2}{\tau}\Sigma^{-1})}\beta - \beta^T\underline{\mathbf{H}^T\mathbf{y}} - \mathbf{y}^T\mathbf{H}\beta\right]$$

- **Note.** $(\beta-\mu)^T\Sigma^{-1}(\beta-\mu) = \beta^T\underline{\Sigma}^{-1}\beta - \beta^T\underline{\Sigma^{-1}\mu} - \mu^T\Sigma^{-1}\beta + \mathbf{A}$

- Thus, the posterior for β is also gaussian with

$$\mathbb{E}(\beta\,|\,\mathbf{Z}) = \left(\mathbf{H}^T\mathbf{H} + \frac{\sigma^2}{\tau}\sigma^{-1}\right)^{-1}\mathbf{H}^T\mathbf{y},\ cov(\beta\,|\,\mathbf{Z}) = \left(\mathbf{H}^T\mathbf{H} + \frac{\sigma^2}{\tau}\sigma^{-1}\right)^{-1}\sigma^2$$

Bayesian Methods

- Corresponding posterior values for $\mu(x)$,

$$\mathbb{E}(\mu(x)\,|\,\mathbf{Z}) = h(x)^T\left(\mathbf{H}^T\mathbf{H} + \frac{\sigma^2}{\tau}\Sigma^{-1}\right)^{-1}\mathbf{H}^T\mathbf{y},\; cov(\mu(x), \mu(x')\,|\,\mathbf{Z}) = h(x)^T\left(\mathbf{H}^T\mathbf{H} + \frac{\sigma^2}{\tau}\Sigma^{-1}\right)^{-1} h(x')\sigma^2$$

- Let $\sigma^2 = \hat{\sigma}^2$, thus enough to choose the prior Σ, τ. The prior can be chosen from subject matter knowledge about the parameters. Here we are willing to say $\mu(x)$ should be smooth, and have guaranteed this by expressing μ in as smooth low-dimensional basis of B-splines. Hence we can take $\Sigma = \mathbf{I}$.

- As prior variance $\tau \to \infty$, the posterior and the bootstrap distribution coincide. In chapter 3, we show that MAP assuming a normal distribution for the prior β is equivalent to Ridge.

Bayesian Methods

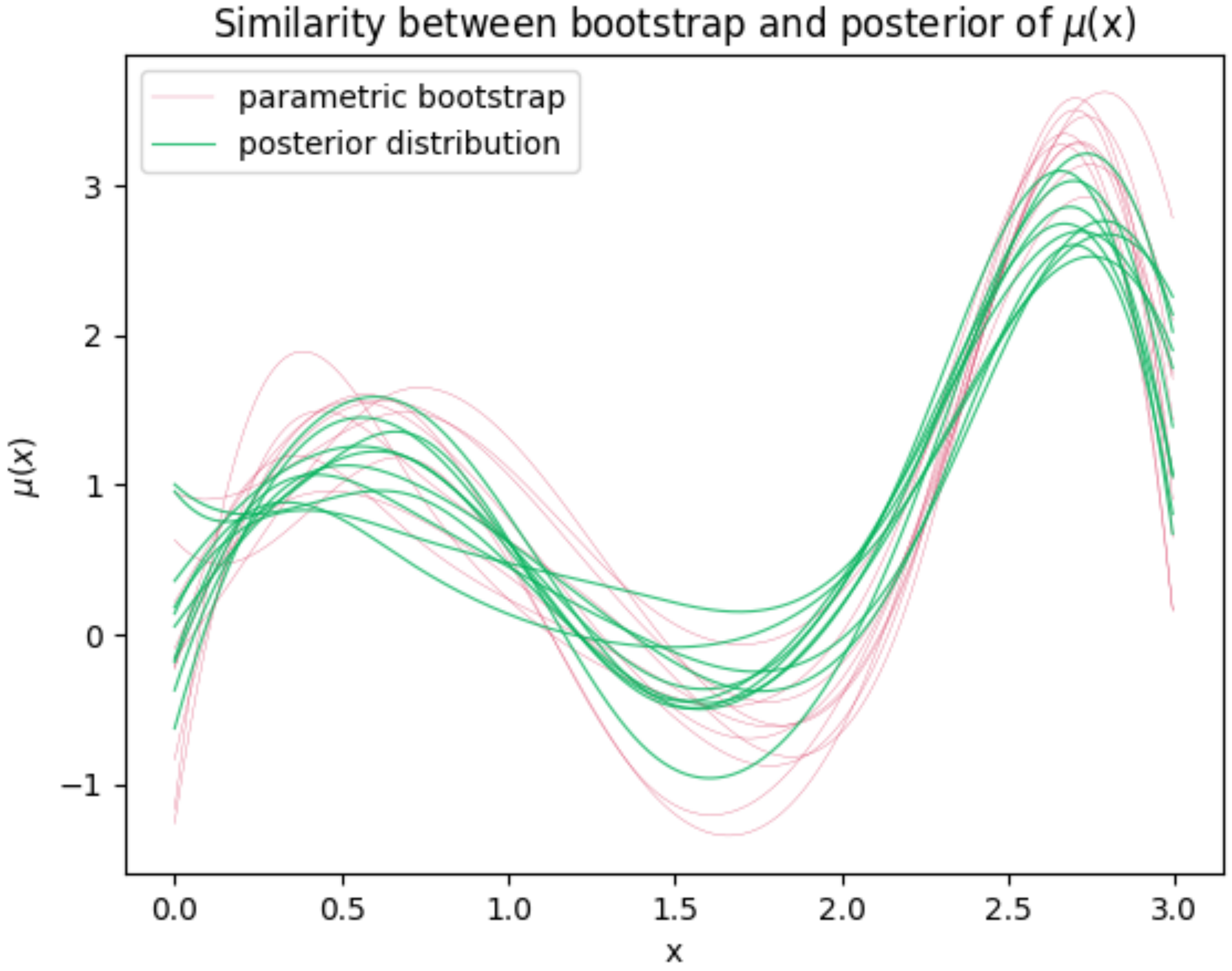
Bootstrap distribution (Maximum likelihood)

$$\beta \sim \mathcal{N}\left(\hat{\beta}^{lse}, \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\hat{\sigma}^2\right)$$

Posterior distribution

$$\beta \sim \mathcal{N}\left(\hat{\beta}^{ridge}, (\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I})^{-1}\sigma^2\right) \text{ where } \lambda = \frac{\sigma^2}{\tau}$$

As $\tau \to \infty$, $\lambda \to 0$ is called a non-informative prior for θ. Then the posterior distribution and the bootstrap distribution coincide.



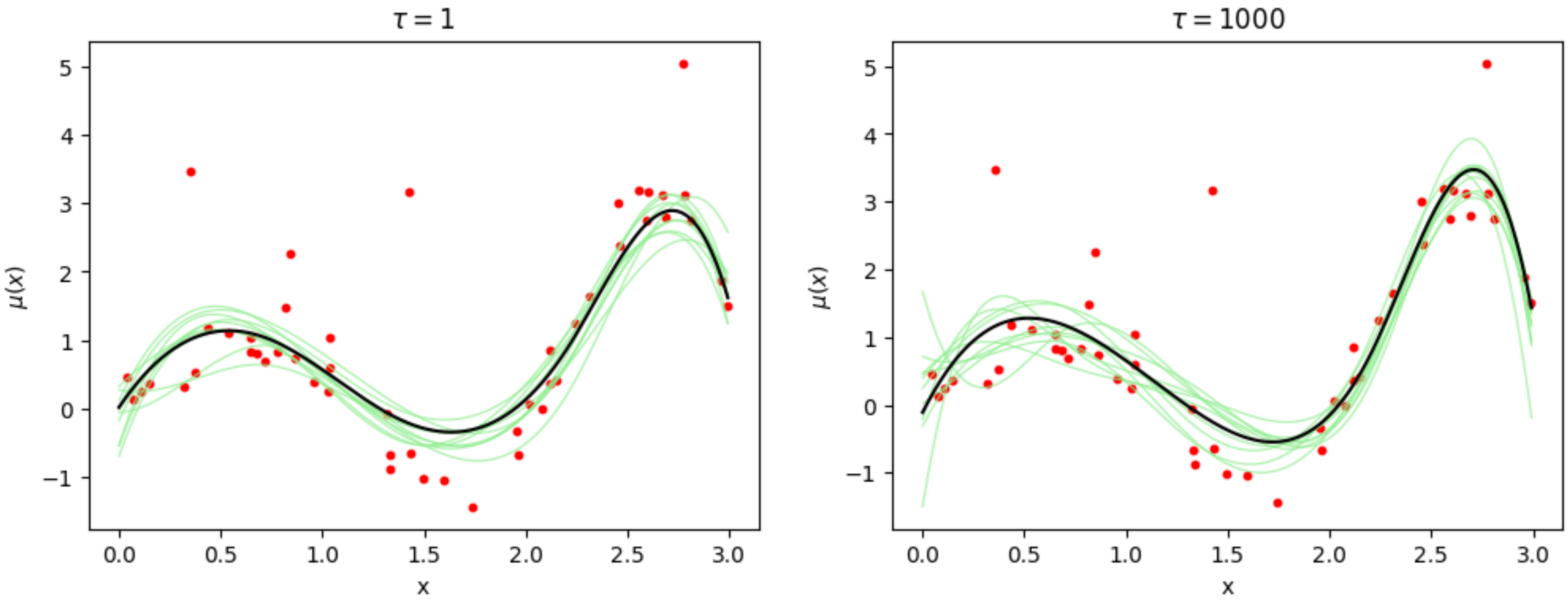
We have used a non informative prior for σ^2 and replaced it with the maximum likelihood estimate $\hat{\sigma}^2$ in the posterior.

A more standard Bayesian analysis would also put a prior on σ, calculate a joint posterior for $\mu(x)$ and σ, and then integrate out σ.

Bayesian Methods

Generates 10 posterior β' from $\beta \sim \mathcal{N}\left(\hat{\beta}^{ridge}, (\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I})^{-1}\sigma^2\right)$, giving $\mu'(x) = \sum_1^7 \beta_j' h_j(x)$.

For $\tau = 1$, the posterior curves in the left panel are smoother than the bootstrap curves, because we have imposed more prior weight on smoothness.



Bayesian Methods

- The bootstrap distribution represents an approximate non-parametric, non-informative posterior distribution of for our parameter.

- Bootstrap distribution is obtained painlessly - without having to formally specify a prior and without having to sample from the posterior i.e. is typically much simpler to carry out.

- Hence we might think of the bootstrap distribution as a **“poor man’s” Bayes posterior**.

MCMC for sampling posteriors

- Having defined a Bayesian model, one would like to **draw samples from the resulting posterior distribution**, in order to make inferences about the parameters.
- Except for simple models, this is often a difficult computational problem or sampling directly from posterior is difficult.
- Markov chain Monte Carlo(MCMC) allows sampling from a large class of distributions, and which scales well with the dimensionality of the sample space.

Markov Chain

- A first-order Markov chain is defined to be a series of random variables $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(M)}$ s.t. $p(\mathbf{z}^{(m+1)} \,|\, \mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}) = p(\mathbf{z}^{(m+1)} \,|\, \mathbf{z}^{(m)}), \forall m$

- We can then specify the Markov chain by density for initial variable $p(\mathbf{z}^{(0)})$ together with the transition probabilities $T_m(\mathbf{z}^{(m)}, \mathbf{z}^{(m+1)}) = p(\mathbf{z}^{(m+1)} \,|\, \mathbf{z}^{(m)})$. A Markov chain is called **homogeneous** if T_m are the same for all m.

- A density p is called **stationary** or **invariant** with respect Markov chain if $p(\mathbf{z}^{(m+1)}) = \sum_{\mathbf{z}^{(m)}} T_m(\mathbf{z}^{(m+1)}, \mathbf{z}^{(m)}) p(\mathbf{z}^{(m)}), \forall m$

- **(detailed balance)** $p^*(\mathbf{z})T(\mathbf{z}, \mathbf{z}') = p^*(\mathbf{z}')T(\mathbf{z}', \mathbf{z})$ for particular density p^*. If p^* holds detailed balance condition, then it's invariant. A Markov chain that respects detailed balance is said to be **reversible.**

- A Markov chain is called **ergodic** if $p(\mathbf{z}^{(m)})$ converges to the invariant $p^*(\mathbf{z})$ as $m \to \infty$, irrespective of the choice of the initial distribution $p(\mathbf{z}^{(0)})$. And then the invariant $p^*(\mathbf{z})$ is called the **equilibrium** distribution.

Pattern Recognition and Machine Learning - Bishop

Markov Chain

- Our goal is to use Markov chains to sample from a given distribution. We can achieve this if we set up a Markov chain such that **the desired distribution is invariant**.

- **A homogeneous Markov chain** will be ergodic, subject only to weak restrictions on the **invariant distribution** and the transition probabilities. And an ergodic Markov chain can have only one equilibrium distribution.

- Hence, in practice, it’s enough to check that **detailed balance** property.

Pattern Recognition and Machine Learning - Bishop

The Metropolis-Hastings algorithm

- Let $p(\mathbf{z})$ be target distribution with $p(\mathbf{z}) = \tilde{p}(\mathbf{z})/Z_p$ where $\tilde{p}$ can readily be evaluated for any given value of $\mathbf{z}$, although Z_p may be unknown.

1. Define the proposal distributions $q_k(\mathbf{z} \,|\, \mathbf{z}')$ s.t. straightforward to draw samples from it directly, and initialize $\mathbf{z}^{(0)}$. Here k labels the members of the set of possible transitions being considered.

2. For each iteration τ, we draw a sample $\mathbf{z}^* \sim q_k(\mathbf{z}^* \,|\, \mathbf{z}^{(\tau)})$, and then compute

   $$A_k(\mathbf{z}^*, \mathbf{z}^{(\tau)}) = \min\left(1, \frac{\tilde{p}(\mathbf{z}^*) q_k(\mathbf{z}^{(\tau)} \,|\, \mathbf{z}^*)}{\tilde{p}(\mathbf{z}^{(\tau)}) q_k(\mathbf{z}^* \,|\, \mathbf{z}^{(\tau)})}\right) > u \text{ where } u \sim Unif(0,1)$$

   If above statement is true, then $\mathbf{z}^{(\tau+1)} = \mathbf{z}^*$ else $\mathbf{z}^{(\tau+1)} = \mathbf{z}^{(\tau)}$

Pattern Recognition and Machine Learning - Bishop

The Metropolis-Hastings algorithm

- **(Detailed balance)**

$$p(\mathbf{z})q_k(\mathbf{z}\,|\,\mathbf{z}')A_k(\mathbf{z}',\mathbf{z}) = \min\left(p(\mathbf{z})q_k(\mathbf{z}\,|\,\mathbf{z}'), p(\mathbf{z}')q_k(\mathbf{z}'\,|\,\mathbf{z})\right)$$

$$= \min\left(p(\mathbf{z}')q_k(\mathbf{z}'\,|\,\mathbf{z}), p(\mathbf{z})q_k(\mathbf{z}\,|\,\mathbf{z}')\right) = p(\mathbf{z}')q_k(\mathbf{z}'\,|\,\mathbf{z})A_k(\mathbf{z},\mathbf{z}')$$ as required.

- For continuous state spaces, a common choice is a Gaussian centered on the current state, leading to an important trade-off in determining the variance parameter of this distribution.

- If the variance is small, then the proportion of accepted transitions will be high, but a slow random walk leading to long correlation times.

Gibbs sampling

- Gibbs sampling can be seen as a special case of the Metropolis-Hastings algorithm.

- We have random variables $\mathbf{z} = (z_1, \ldots, z_M)$ with distribution $p(\mathbf{z})$ and difficult to sample from it.

- Gibbs sampling replaces z_i by a value drawn from $p(z_i, \mathbf{z}_{\backslash i}), \forall i = 1,\ldots,M$ where $\mathbf{z}_{\backslash i}$ denotes $\mathbf{z} \backslash \{z_i\}$

1. Initialize $\mathbf{z}^{(1)} = (z_1^{(1)}, \ldots, z_M^{(1)})$

2. For $\tau = 1,\ldots T$:

   Sample $z_i^{(\tau+1)} \sim p(z_i \,|\, \mathbf{z}_{\backslash i}^{(\tau)})$ for $i = 1,\ldots,M$

Pattern Recognition and Machine Learning - Bishop

Gibbs sampling

- Consider a Metropolis-Hastings sampling step involving z_k in which $\mathbf{z}_{\setminus k}$ remain fixed, and for which the transition probability from $\mathbf{z}$ to $\mathbf{z}^*$ is given by $q_k(\mathbf{z}^* \,|\, \mathbf{z}) = p(z_k^* \,|\, \mathbf{z}_{\setminus k})$.

- Since $\mathbf{z}_{\setminus k}$ is fixed, $\mathbf{z}^*_{\setminus k} = \mathbf{z}_{\setminus k}$. And $p(\mathbf{z}) = p(z_k \,|\, \mathbf{z}_{\setminus k})p(\mathbf{z}_{\setminus k})$. Thus the acceptance probability is

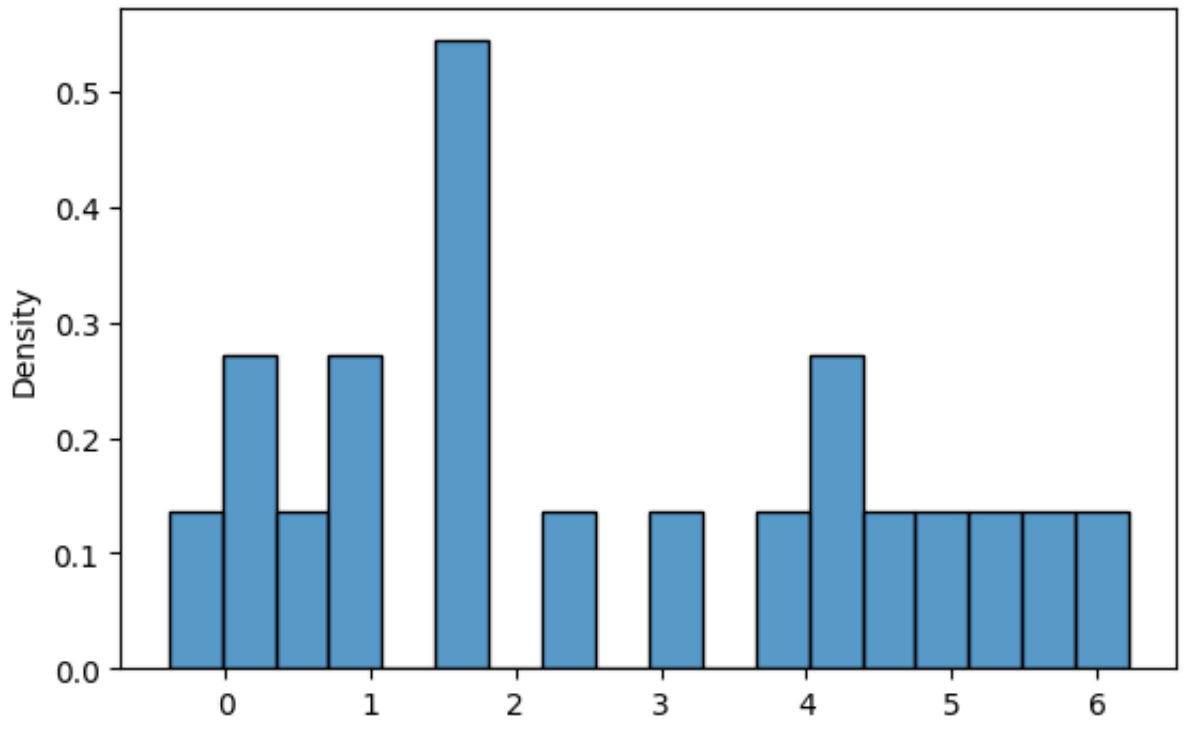
$$A(\mathbf{z}^*, \mathbf{z}) = \frac{p(\mathbf{z}^*)q_k(\mathbf{z} \,|\, \mathbf{z}^*)}{p(\mathbf{z})q_k(\mathbf{z}^* \,|\, \mathbf{z})} = \frac{p(z_k^* \,|\, \mathbf{z}^*_{\setminus k})p(\mathbf{z}^*_{\setminus k})p(z_k \,|\, \mathbf{z}^*_{\setminus k})}{p(z_k \,|\, \mathbf{z}_{\setminus k})p(\mathbf{z}_{\setminus k})p(z_k^* \,|\, \mathbf{z}_{\setminus k})} = 1$$ i.e. always accepted.

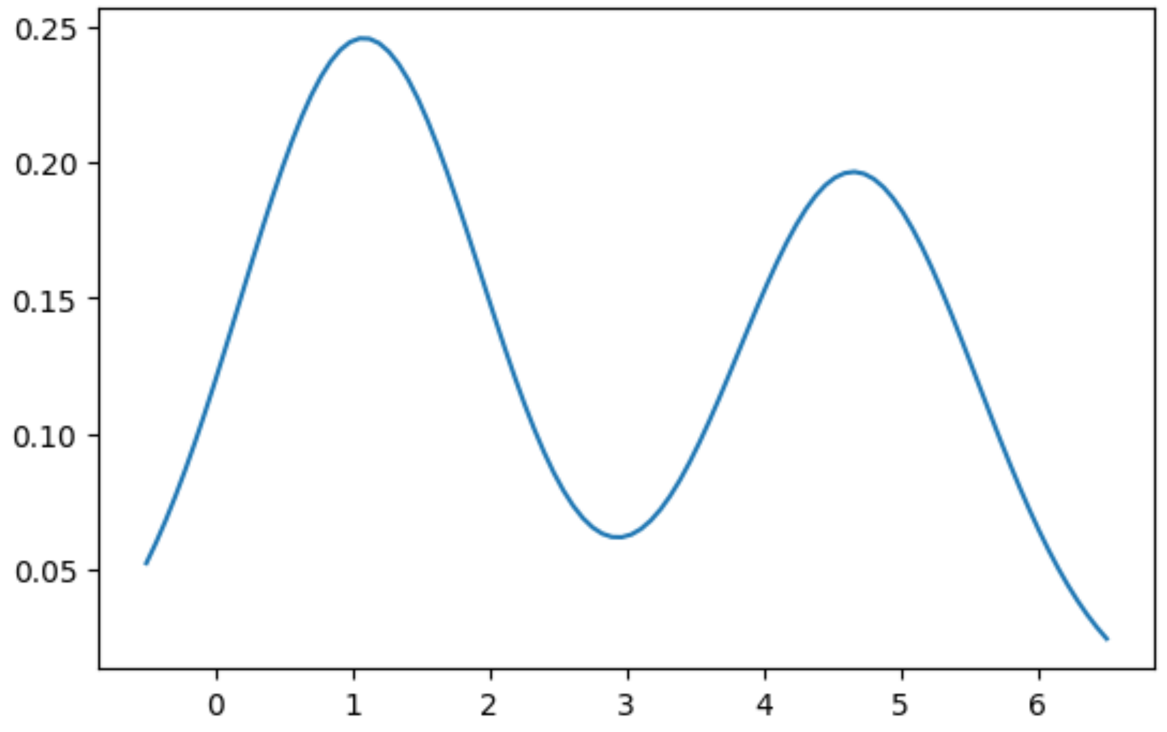
Pattern Recognition and Machine Learning - Bishop

Gibbs sampling

Two-component mixture example:

-0.39	0.12	0.94	1.67	1.76	2.44	3.72	4.28	4.92	5.53
0.06	0.48	1.01	1.68	1.80	3.25	4.12	4.60	5.28	6.22





Model

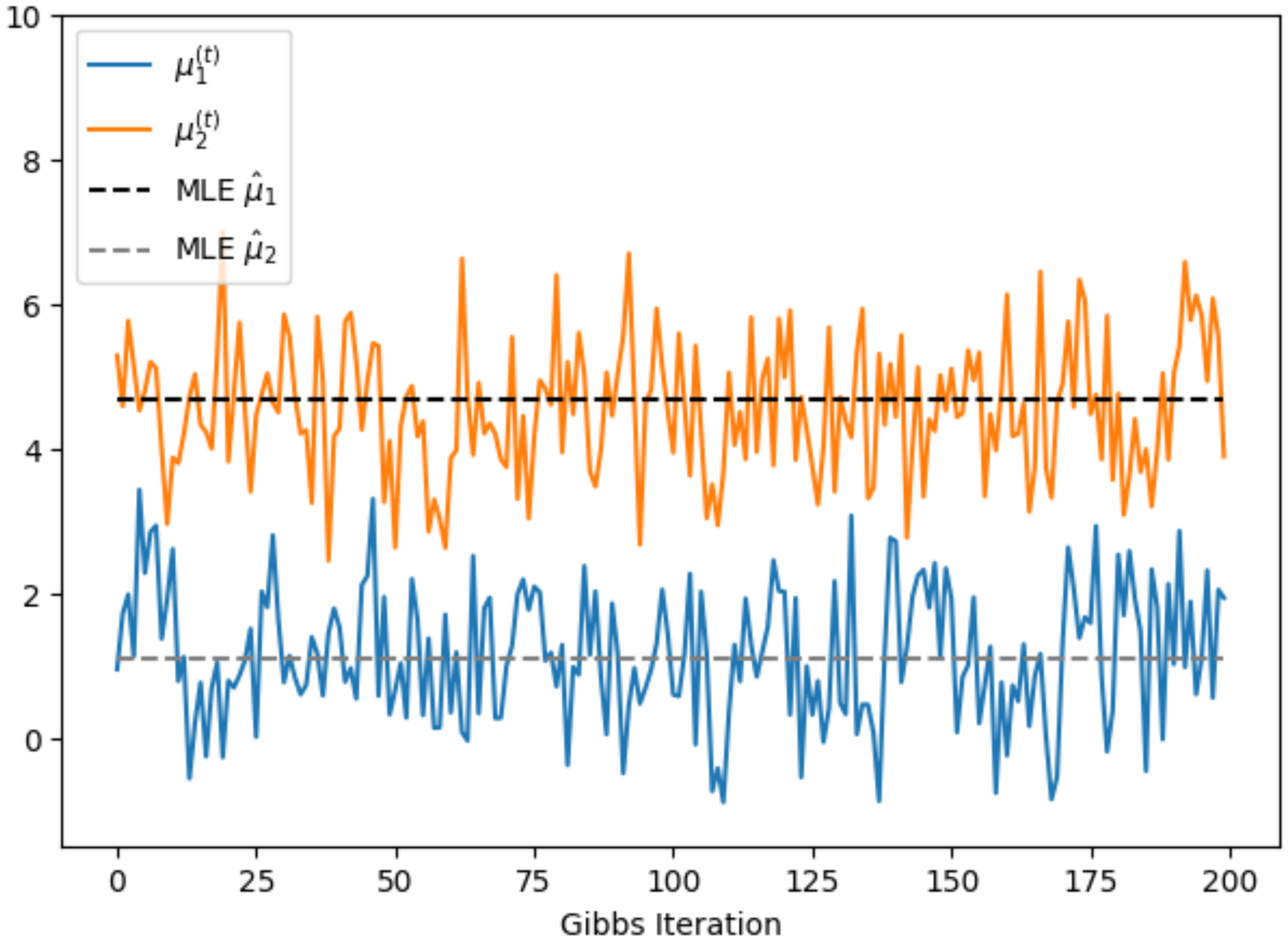
$Y = (1 - \Delta) \cdot Y_1 + \Delta \cdot Y_2$ where $Y_i \sim \mathcal{N}(\mu_i, \sigma_i^2), i = 1{,}2,$ $\Delta \in \{0{,}1\}$ with $Pr(\Delta = 1) = \pi$. Since Δ is unknown, substituting for each Δ_i for $\gamma_i = \mathbb{E}(\Delta_i | \theta, \mathbf{Z}) = Pr(\Delta_i = 1 | \theta, \mathbf{Z})$

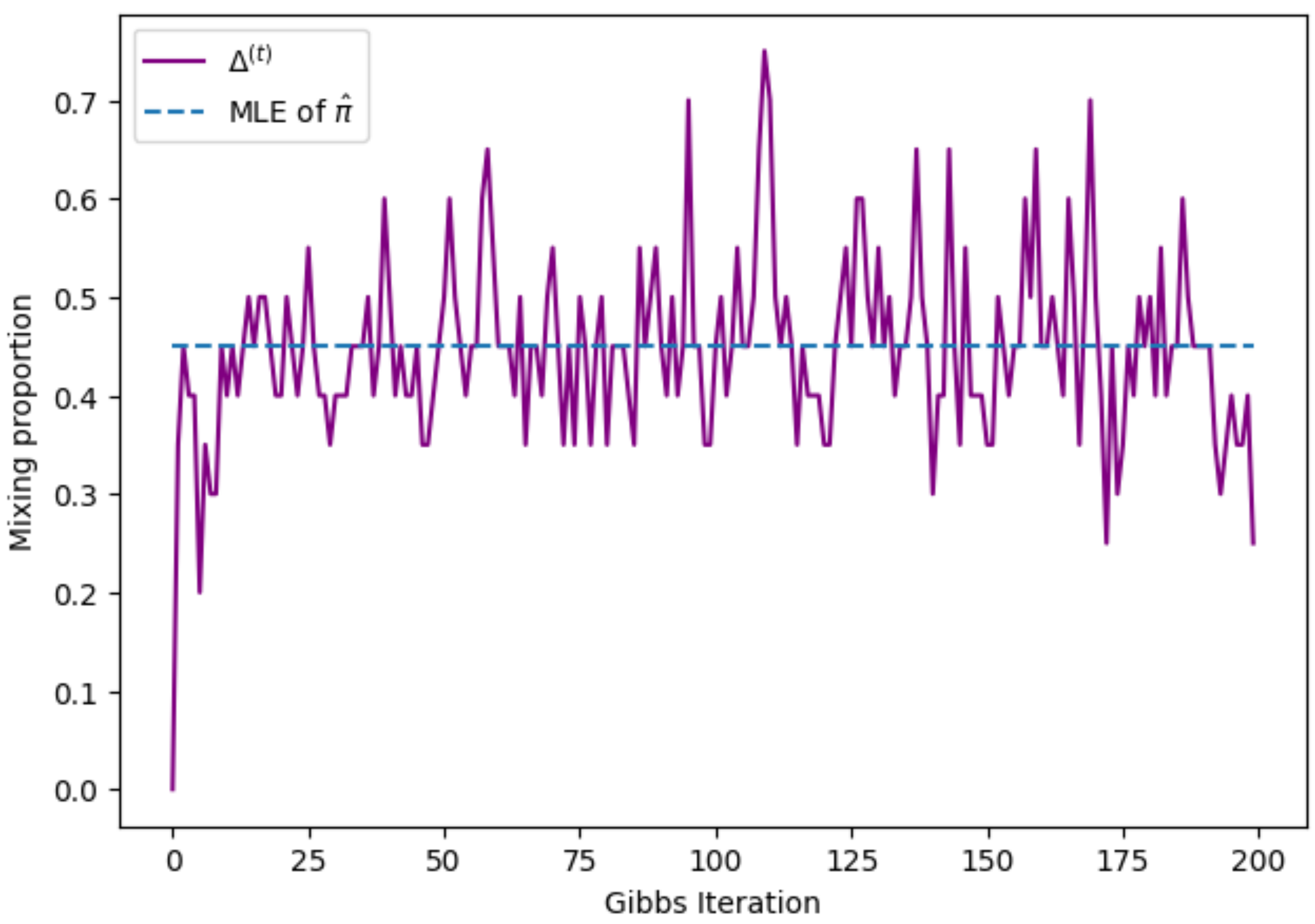
$$\hat{\gamma}_i = \frac{\hat{\pi}\phi_{\hat{\theta}_2}(y_i)}{(1-\hat{\pi})\phi_{\hat{\theta}_1}(y_i) + \hat{\pi}\phi_{\hat{\theta}_2}(y_i)}, \quad i = 1, 2, \ldots, N. \qquad (8.42)$$

Algorithm 8.4 *Gibbs sampling for mixtures.*

```
1. Take some initial values $\theta^{(0)} = (\mu_1^{(0)}, \mu_2^{(0)})$.
2. Repeat for $t = 1, 2, \ldots, .$
    (a) For $i = 1, 2, \ldots, N$ generate $\Delta_i^{(t)} \in \{0, 1\}$ with $\Pr(\Delta_i^{(t)} = 1) = \hat{\gamma}_i(\theta^{(t)})$, from equation (8.42).
    (b) Set
            $\hat{\mu}_1 = \frac{\sum_{i=1}^N (1 - \Delta_i^{(t)}) \cdot y_i}{\sum_{i=1}^N (1 - \Delta_i^{(t)})},$
            $\hat{\mu}_2 = \frac{\sum_{i=1}^N \Delta_i^{(t)} \cdot y_i}{\sum_{i=1}^N \Delta_i^{(t)}},$
        and generate $\mu_1^{(t)} \sim N(\hat{\mu}_1, \hat{\sigma}_1^2)$ and $\mu_2^{(t)} \sim N(\hat{\mu}_2, \hat{\sigma}_2^2)$.
3. Continue step 2 until the joint distribution of $(\mathbf{\Delta}^{(t)}, \mu_1^{(t)}, \mu_2^{(t)})$ doesn't change
```

Gibbs sampling





The EM Algorithm

- The EM algorithm is designed to compute the MLE or MAP parameter estimate for probability models that have missing data and/or hidden variables.

- The basic idea behind EM is to alternate between estimating the hidden variables (or missing values) during the **E step (expectation step)**, and then using the fully observed data to compute the MLE during the **M step (maximization step)**.

- The goal of EM is to maximize the log likelihood of the observed data $\mathbf{Z}$ with latent or missing data $\mathbf{Z}^m$.

- Suppose that direct optimization of $Pr(\mathbf{Z} \,|\, \theta)$ is difficult, but that optimization of the complete-data likelihood $Pr(\mathbf{T} \,|\, \theta) = Pr(\mathbf{Z}, \mathbf{Z}^m \,|\, \theta)$ is significantly easier.

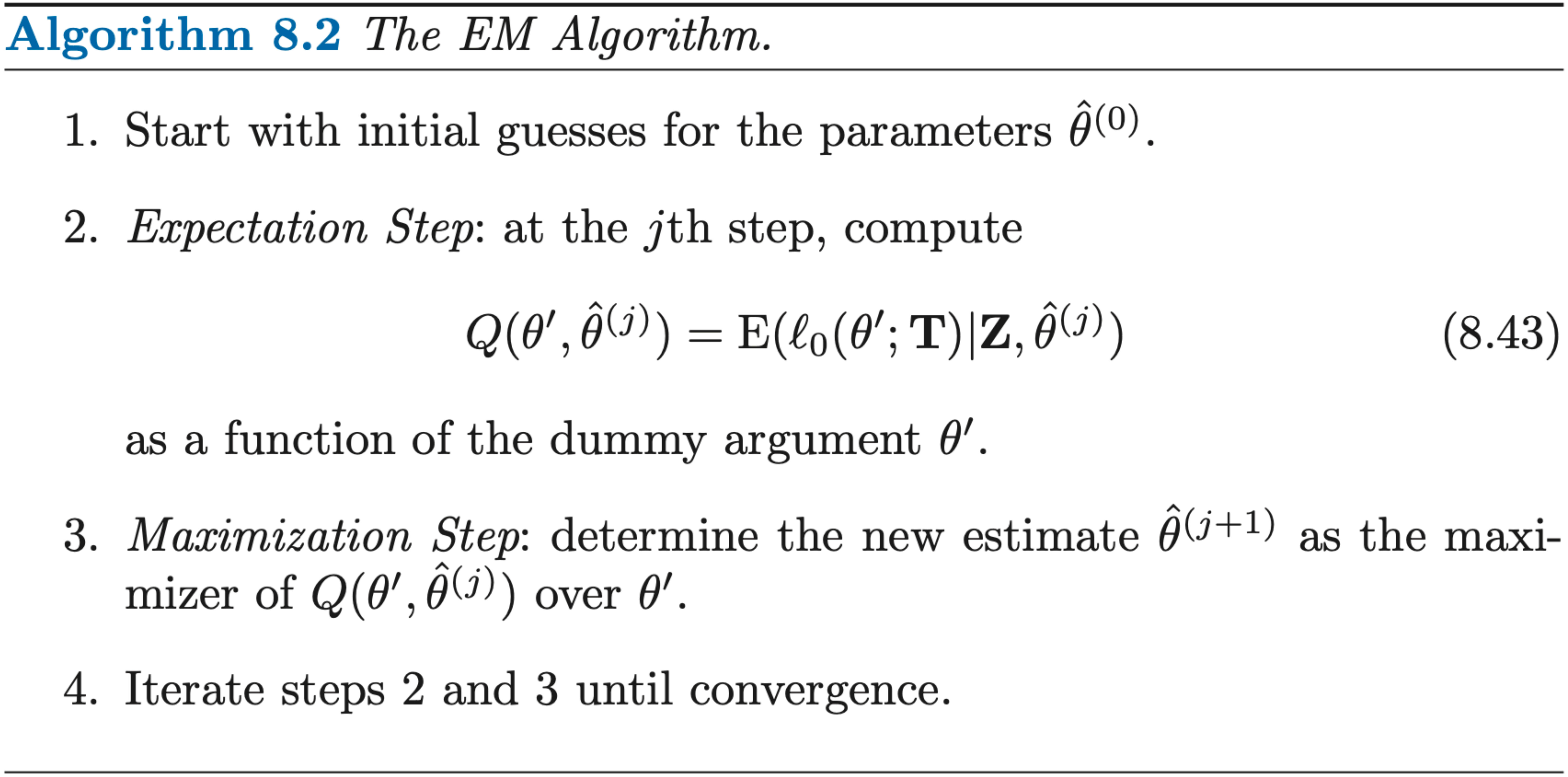
The EM Algorithm

- Since $Pr(\mathbf{Z}^m \,|\, \mathbf{Z}, \theta') = \dfrac{Pr(\mathbf{T} \,|\, \theta')}{Pr(\mathbf{Z} \,|\, \theta')}$, $\log Pr(\mathbf{Z} \,|\, \theta') = \log Pr(\mathbf{T} \,|\, \theta') - \log Pr(\mathbf{Z}^m \,|\, \mathbf{Z}, \theta')$
- Then taking conditional expectations with respect to $\mathbf{T} \,|\, \mathbf{Z}$ governed by θ i.e. $Pr(\mathbf{Z}^m \,|\, \mathbf{Z}, \theta)$

$$\begin{aligned}\mathbb{E}_{Pr(\mathbf{Z}^m|\mathbf{Z},\theta)} \log Pr(\mathbf{Z} \,|\, \theta') &= \int \log Pr(\mathbf{Z} \,|\, \theta') Pr(z^m \,|\, \mathbf{Z}, \theta) dz^m = \log Pr(\mathbf{Z} \,|\, \theta') \cdot 1 \\ &= \mathbb{E}_{Pr(\mathbf{Z}^m|\mathbf{Z},\theta)} \log Pr(\mathbf{T} \,|\, \theta') - \mathbb{E}_{Pr(\mathbf{Z}^m|\mathbf{Z},\theta)} \log Pr(\mathbf{Z}^m \,|\, \mathbf{Z}, \theta') \\ &=: Q(\theta', \theta) - R(\theta', \theta) =: \mathscr{L}(\theta')\end{aligned}$$

- Thus, $l(\theta'; \mathbf{Z}) = \mathbb{E}_{Pr(\mathbf{Z}^m|\mathbf{Z},\theta)} \log Pr(\mathbf{Z} \,|\, \theta') = \mathscr{L}(\theta')$, our goal is to find $\underset{\theta'}{max} \mathscr{L}(\theta')$ given θ.

The EM Algorithm



Algorithm 8.2 *The EM Algorithm.*

1. Start with initial guesses for the parameters $\hat{\theta}^{(0)}$.
2. *Expectation Step*: at the jth step, compute

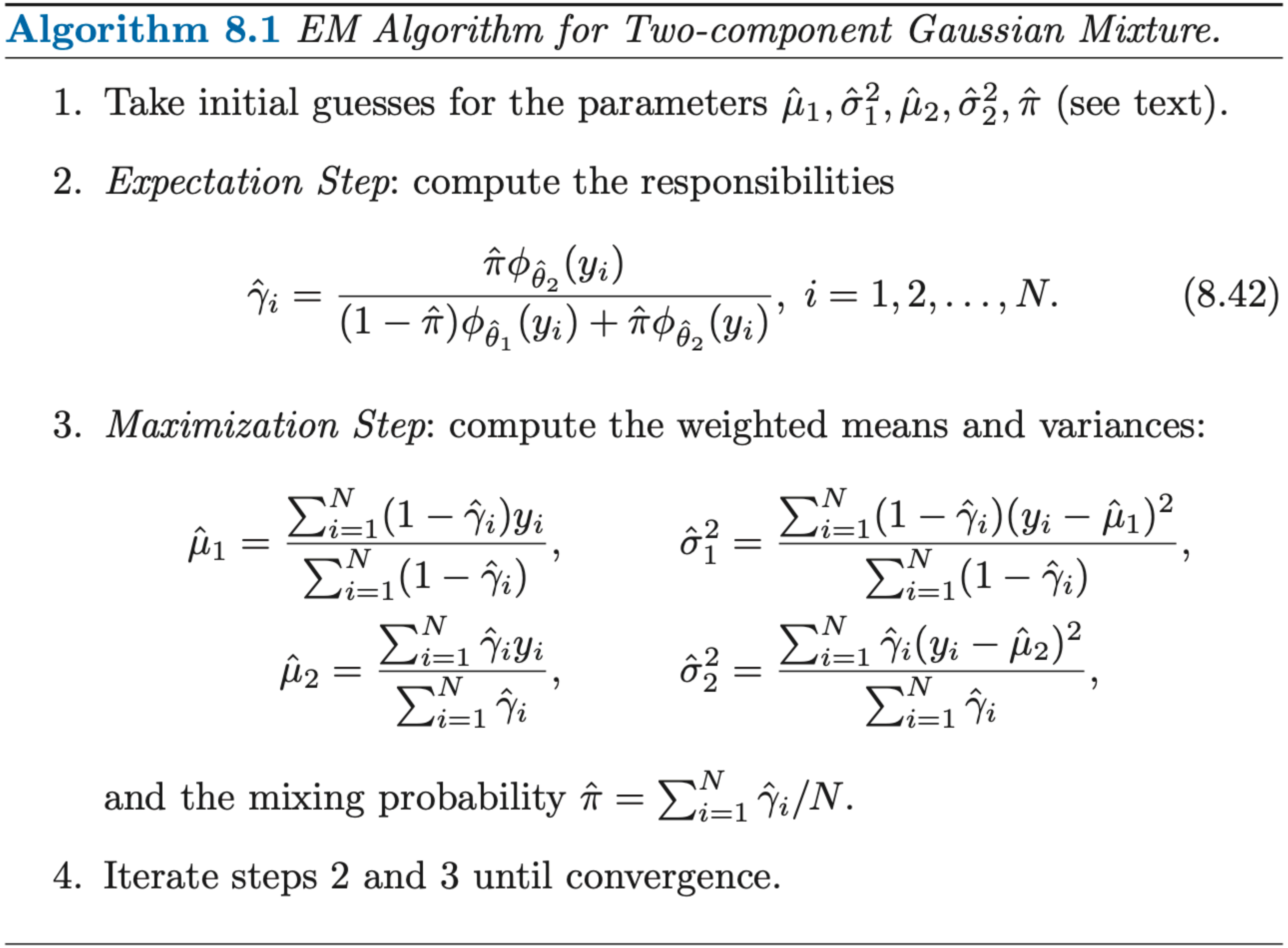
$$Q(\theta', \hat{\theta}^{(j)}) = \mathrm{E}(\ell_0(\theta'; \mathbf{T})|\mathbf{Z}, \hat{\theta}^{(j)}) \tag{8.43}$$

   as a function of the dummy argument θ'.

3. *Maximization Step*: determine the new estimate $\hat{\theta}^{(j+1)}$ as the maximizer of $Q(\theta', \hat{\theta}^{(j)})$ over θ'.
4. Iterate steps 2 and 3 until convergence.

The EM Algorithm

Mixture example in Gibbs sampling: Initial guess for $\hat{\mu}_1, \hat{\mu}_2$ is to choose two of the y_i at random, $\hat{\sigma}_1^2 = \hat{\sigma}_2^2 = \frac{1}{N}\sum (y_i - \bar{y})^2$, $\hat{\pi}_1 = \hat{\pi}_2 = 0.5$



Algorithm 8.1 *EM Algorithm for Two-component Gaussian Mixture.*

1. Take initial guesses for the parameters $\hat{\mu}_1, \hat{\sigma}_1^2, \hat{\mu}_2, \hat{\sigma}_2^2, \hat{\pi}$ (see text).
2. *Expectation Step*: compute the responsibilities

$$\hat{\gamma}_i = \frac{\hat{\pi}\phi_{\hat{\theta}_2}(y_i)}{(1-\hat{\pi})\phi_{\hat{\theta}_1}(y_i) + \hat{\pi}\phi_{\hat{\theta}_2}(y_i)}, \; i = 1, 2, \ldots, N. \qquad (8.42)$$

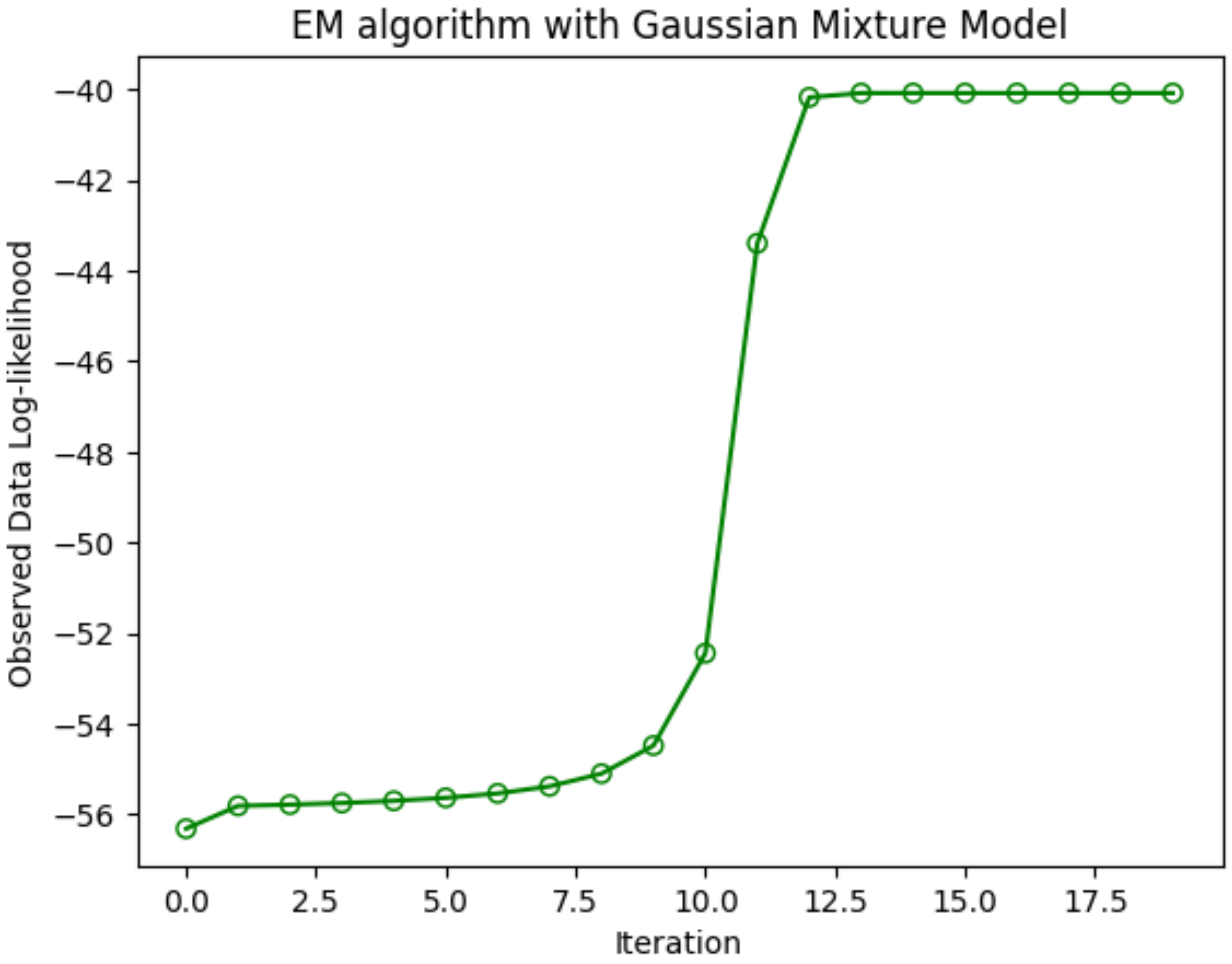
3. *Maximization Step*: compute the weighted means and variances:

$$\hat{\mu}_1 = \frac{\sum_{i=1}^N (1-\hat{\gamma}_i) y_i}{\sum_{i=1}^N (1-\hat{\gamma}_i)}, \qquad \hat{\sigma}_1^2 = \frac{\sum_{i=1}^N (1-\hat{\gamma}_i)(y_i - \hat{\mu}_1)^2}{\sum_{i=1}^N (1-\hat{\gamma}_i)},$$

$$\hat{\mu}_2 = \frac{\sum_{i=1}^N \hat{\gamma}_i y_i}{\sum_{i=1}^N \hat{\gamma}_i}, \qquad \hat{\sigma}_2^2 = \frac{\sum_{i=1}^N \hat{\gamma}_i (y_i - \hat{\mu}_2)^2}{\sum_{i=1}^N \hat{\gamma}_i},$$

   and the mixing probability $\hat{\pi} = \sum_{i=1}^N \hat{\gamma}_i / N$.

4. Iterate steps 2 and 3 until convergence.



The EM Algorithm

- **(Theorem)** If θ^* maximizes $Q(\theta', \theta)$, then $\mathscr{L}(\theta^*) \geq \mathscr{L}(\theta)$. Hence the EM iteration never decreases the log-likelihood.

  Proof. Then $Q(\theta^*, \theta) - Q(\theta, \theta) \geq 0$. Thus, enough to check that $R(\theta^*, \theta) \leq R(\theta, \theta)$.

  Then $R(\theta^*, \theta) - R(\theta, \theta) = \mathbb{E}_{Pr(\mathbf{Z}^m|\theta)} \log \dfrac{Pr(\mathbf{Z}^m \,|\, \mathbf{Z}, \theta^*)}{Pr(\mathbf{Z}^m \,|\, \mathbf{Z}, \theta)} \leq \mathbb{E}_{Pr(\mathbf{Z}^m|\theta)} \log \dfrac{Pr(\mathbf{Z}^m \,|\, \mathbf{Z}, \theta)}{Pr(\mathbf{Z}^m \,|\, \mathbf{Z}, \theta)} = \log 1 = 0$ by Jensen's inequality. Therefore, $R(\theta^*, \theta) - R(\theta, \theta) \leq 0$.

- This argument also makes it cleat that a full maximization in the M step is not necessary. We need only find $\hat{\theta}^{(j+1)}$ s.t. $Q(\hat{\theta}^{(j+1)}, \hat{\theta}^{(j)}) > Q(\hat{\theta}^{(j)}, \hat{\theta}^{(j)})$. Such procedures are called GEM(generalized EM) algorithms.

Generalized EM

- We merely increase the expected complete data log-likelihood, rather than maximizing it. For example, we might follow a few gradient steps.

$\theta_{t+1} = \theta_t - \eta_t \mathbf{H}_t^{-1} \mathbf{g}_t$ where $0 \leq \eta_t \leq 1$ is step size, and $\mathbf{g}_t = \frac{\partial}{\partial \theta} Q(\theta, \theta_t) \,|_{\theta=\theta_t}$

$$\mathbf{H}_t = \frac{\partial^2}{\partial \theta \partial \theta^T} Q(\theta, \theta_t) \,|_{\theta=\theta_t}$$

- If $\eta_t = 1$, calls this the gradient EM algorithm. Note, however, when the M step cannot be computed in closed form, **EM loses some of its appeal over directly optimizing the marginal likelihood** with a gradient based solver.

.

Minorize-Maximize algorithms

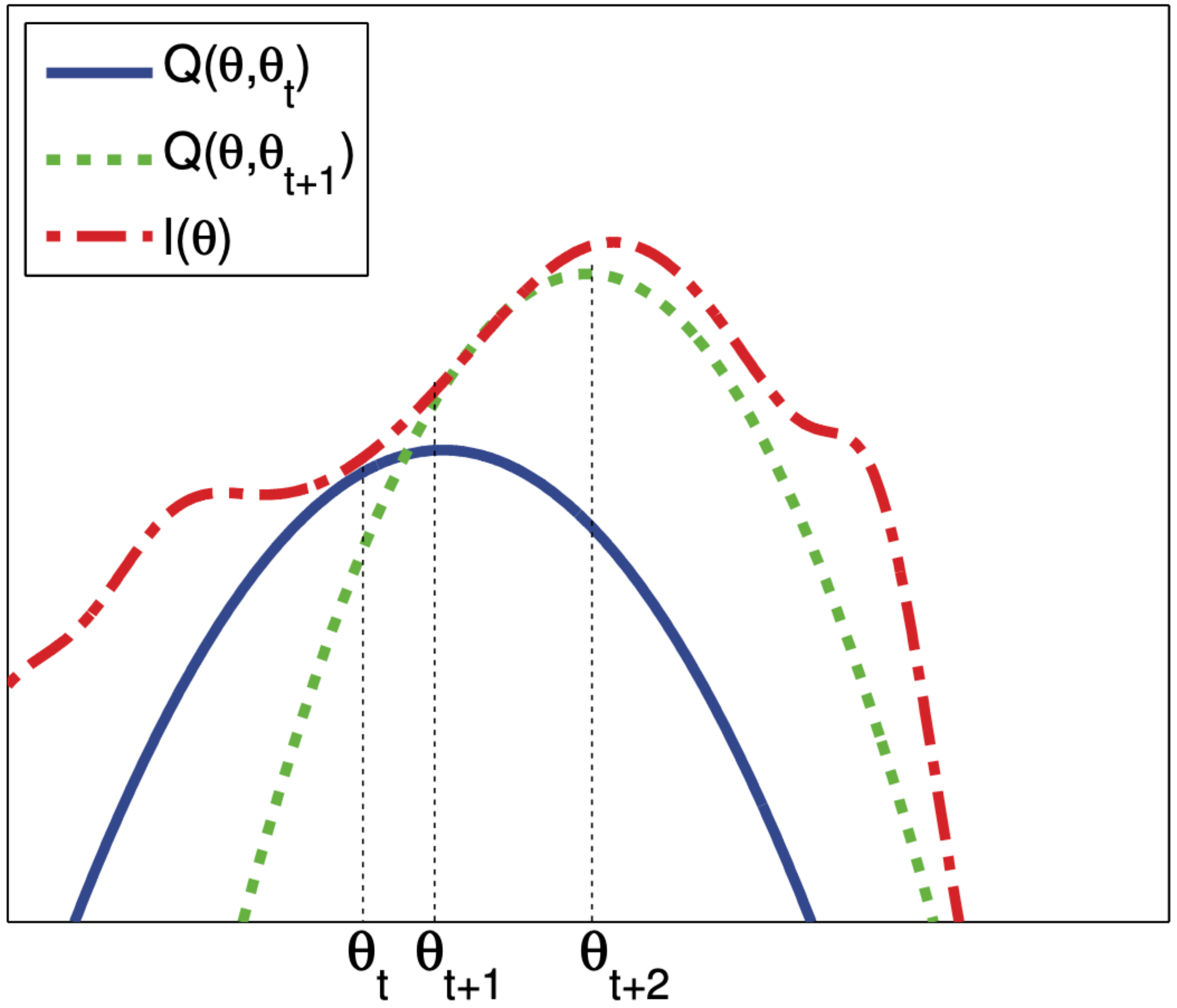
- **(MM algorithms)** we assume our goal is to maximize some function $l(\theta)$ wrt θ. The basic approach in MM algorithms is to construct a **surrogate function** $Q(\theta, \theta^t)$**.**

- A function $Q(\theta, \theta^t)$ is to said to ***minorize* the function** $l(\theta)$ at θ^t provided

$$l(\theta) \geq Q(\theta, \theta^t), \forall \theta, \text{ and } l(\theta^t) = Q(\theta^t, \theta^t)$$

- We then perform $\theta^{t+1} = \underset{\theta}{\text{argmax}} Q(\theta, \theta^t)$ at each step. This guarantees us monotonic increases in the original objective: $l(\theta^{t+1}) \geq Q(\theta^{t+1}, \theta^t) \geq Q(\theta^t, \theta^t) = l(\theta^t)$

Rationale for the MM Principle



- It can generate an algorithm that avoids matrix inversion.
- It can separate the parameters of a problem.
- It can linearize an optimization problem.
- It can deal gracefully with equality and inequality constraints.
- It can turn a non-differentiable problem into a smooth problem

Example for MM algorithm: logistic regression

- If $l(\theta)$ is a concave function we want to maximize, then one way to obtain a valid lower bound is to use a its Hessian in Taylor expansion i.e. to find a negative definite matrix $\mathbf{B}$ s.t. $\mathbf{B} \prec \mathbf{H}(\theta)$
- Thus, $l(\theta) \geq l(\theta^t) + (\theta - \theta^t)^T \nabla l(\theta^t) + \frac{1}{2}(\theta - \theta^t)^T \mathbf{B}(\theta - \theta^t)$. And we can find a valid lower bound:

$$Q(\theta, \theta^t) = \theta^t(\nabla l(\theta^t) - \mathbf{B}\theta^t) + \frac{1}{2}\theta^T \mathbf{B}\theta$$

- The corresponding update becomes $\nabla_\theta Q(\theta, \theta^t) = 0$: $\theta^{t+1} = \theta^t - \mathbf{B}^{-1} \nabla l(\theta^t)$
- This is similar to a Newton update, except we use **fixed** $\mathbf{B}$ rather than $\mathbf{H}(\theta^t)$ which changes at each iteration. -> lower computational cost.

Probabilistic Machine Learning : Advanced Topics - K.Murphy

Example for MM algorithm: logistic regression

- We show that $\mathbf{H}(\theta^t) = -\mathbf{X}^T\mathbf{W}\mathbf{X}$ where $\mathbf{W} = diag(p_i(1-p_i))$ in chapter 4. Since $-p_i(1-p_i) \geq -0.25$ for all i, $-\frac{1}{4}\mathbf{X}^T\mathbf{X} \prec \mathbf{H}(\theta^t)$.

- **(IRLS)** $\beta^{t+1} \leftarrow \beta^t + (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y}-\mathbf{p})$

- **(MM algorithms)** $\beta^{t+1} \leftarrow \beta^t + 4(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y}-\mathbf{p})$ which is faster per step than the IRLS, because we can pre-compute the constant matrix $(\mathbf{X}^T\mathbf{X})^{-1}$.

EM as a MM algorithm

- Denote $Q(\theta', \theta) = Q_{EM}(\theta', \theta) + \log Pr(\mathbf{Z} \,|\, \theta) - Q_{EM}(\theta, \theta)$ where $Q_{EM}(\theta', \theta) = \mathbb{E}_{Pr(\mathbf{Z}^m|\mathbf{Z},\theta)} \log Pr(\mathbf{T} \,|\, \theta')$, and $l(\theta') = l(\theta'; \mathbf{Z}) = \log Pr(\mathbf{Z} \,|\, \theta') = Q_{EM}(\theta', \theta) - R_{EM}(\theta', \theta)$
- Then $Q(\theta', \theta)$ minorize the observed data log-likelihood $l(\theta')$.

Proof. $Q(\theta, \theta) = l(\theta), \forall \theta$ is trivial from our definition. Thus, it's enough to check that $l(\theta') \geq Q(\theta', \theta), \forall \theta'$

We know that $R_{EM}(\theta, \theta) \geq R_{EM}(\theta', \theta)$. So, $l(\theta') - l(\theta) \geq Q_{EM}(\theta', \theta) - Q_{EM}(\theta, \theta)$.

Hence, $Q(\theta', \theta) = Q_{EM}(\theta', \theta) + \log Pr(\mathbf{Z} \,|\, \theta) - Q_{EM}(\theta, \theta) \leq l(\theta')$

Bagging

- Consider the regression problem: Suppose we fit a model to $\mathbf{Z} = \{(x_i, y_i)\}_{i=1}^N$, obtaining the prediction $\hat{f}(x)$ at input x.

- Bootstrap aggregation or bootstrap averages this prediction over a collection of bootstrap sample i.e. $\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x)$ where $\hat{f}^{*b}$ is fitted from bootstrap sample $\mathbf{Z}^{*b}, b = 1,\ldots,B$, thereby reducing its variance.

- Choose the best $\hat{f}^{*b}$ instead averaging, is called **Bumping**. Ex: XOR problem

- In fact the "true" bagging estimate is $\mathbb{E}_{\hat{\mathscr{P}}}\hat{f}^*(x)$ where $\mathbf{Z}^* = \{(x_i, y_i)\}_{i=1}^N, (x_i, y_i) \sim \hat{\mathscr{P}}$, $\hat{\mathscr{P}}$ is empirical distribution on (x_i, y_i). Our estimate is a **Monte Carlo estimate of the true bagging estimate**, approaching it as $B \to \infty$.

- The bagged estimate will differ form the original estimate $\hat{f}(x)$ only when the latter is a non-linear or adaptive function of the data.

Bagging

- Consider the K- class classification problem: Our classifier $\hat{G}(x) = \arg\max_k \hat{f}(x)$ where $\hat{f}(x)$ is indicator function s.t. single 1 value, and $(K-1)$ zeroes.

- **(Majority Vote)** The bagged estimate is $\hat{f}_{bag}(x) = [p_1(x), \dots, p_K(x)]$ where $p_k(x)$ is proportions of each classifier predicting class k at x. Then $\hat{G}_{bag}(x) = \arg\max_k \hat{f}_{bag}(x)$. $p_k(x)$ is **NOT the class-probability estimate** at x. Suppose binary classification with true probability of class 1 at x is 0.75, and each of the bagged classifiers accurately predict a class 1.

- **(Average the probability)** If the classifier $\hat{G}(x)$ already exists $\hat{f}(x)$ that estimates the probabilities at x, then average theses instead, rather than voting. It tends to produce bagged classifier with lower variance, especially for small B.

Bagging

- Bagging improves prediction by reducing the variance without changing bias under squared-error loss. Assume $\{(x_i, y_i)\}_{i=1}^{N}$(x is fixed.) are independently drawn form the actual population $\mathscr{P}$, and consider the ideal aggregator $f_{ag}(x) = \mathbb{E}_{\mathscr{P}}\hat{f}^*(x)$. Then, we can write

$$
\begin{aligned}
\mathbb{E}_{\mathscr{P}}[Y - \hat{f}^*(x)]^2 &= \mathbb{E}_{\mathscr{P}}[Y - f_{ag}(x) + f_{ag}(x) - \hat{f}^*(x)]^2 \\
&= \mathbb{E}_{\mathscr{P}}[Y - f_{ag}(x)]^2 + \underbrace{\mathbb{E}_{\mathscr{P}}[f_{ag}(x) - \hat{f}^*(x)]^2}_{\textbf{The variance of } \hat{f}^*(x) \textbf{ around its mean } f_{ag}(x)}(\because f_{ag} \text{is the ideal aggregator}) \\
&\geq \mathbb{E}_{\mathscr{P}}[Y - f_{ag}(x)]^2
\end{aligned}
$$

- This suggests that bagging will often decrease mean-squared errors, but it does not hold for classification under 0-1 loss, because of non-additivity of bias and variance.

Bagging

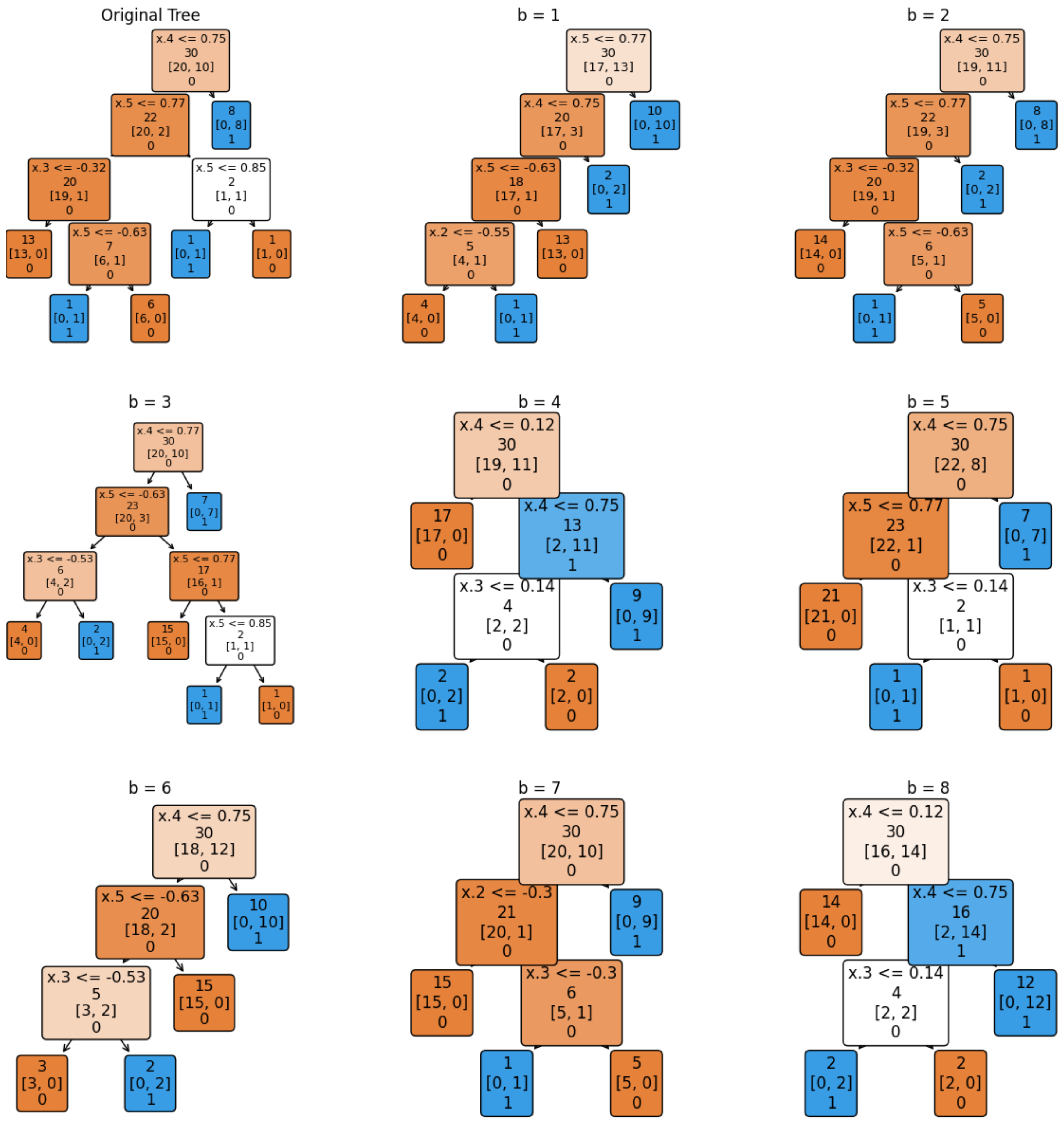
- **(Wisdom of Crowds)** Bagging a good classifier can make it better, but bagging a bad classifier can make it worse. Let the Bayes optimal decision at x be $G(x) = 1$ in binary-classification.

- Suppose each of the **independent** weak learners G_b^* have error-rate $e_b = e < 0.5$, and $S_1(x) = \sum_b I(G_b^*(x) = 1)$ be the consensus vote for class 1. Then $S_1(x) \sim Bin(B, 1 - e)$, $Pr(S_1 > B/2) \to 1 (\because \mathbb{E}[S_1] = B(1 - e) > B/2)$. as $B \to \infty$.

- Note that we bag a model, any simple structure in the model is lost. For interpretation of the model this is clearly a drawback.

Bagging

- **(Example: Trees with Simulated Data)** Generate $N = 30$ samples (x_i, y_i) where $x_i \sim \mathcal{N}_5(0,\Sigma)$, $\Sigma_{ij} = 1$ if $i = j$ else 0.95, and $y_i \sim Bernoulli(0.2)$ if $x_{i1} \leq 0.5$ else $y_i \sim Bernoulli(0.8)$. Thus, the Bayes error rate is 0.2. A test sample of size 2000 was generated from the sample population.

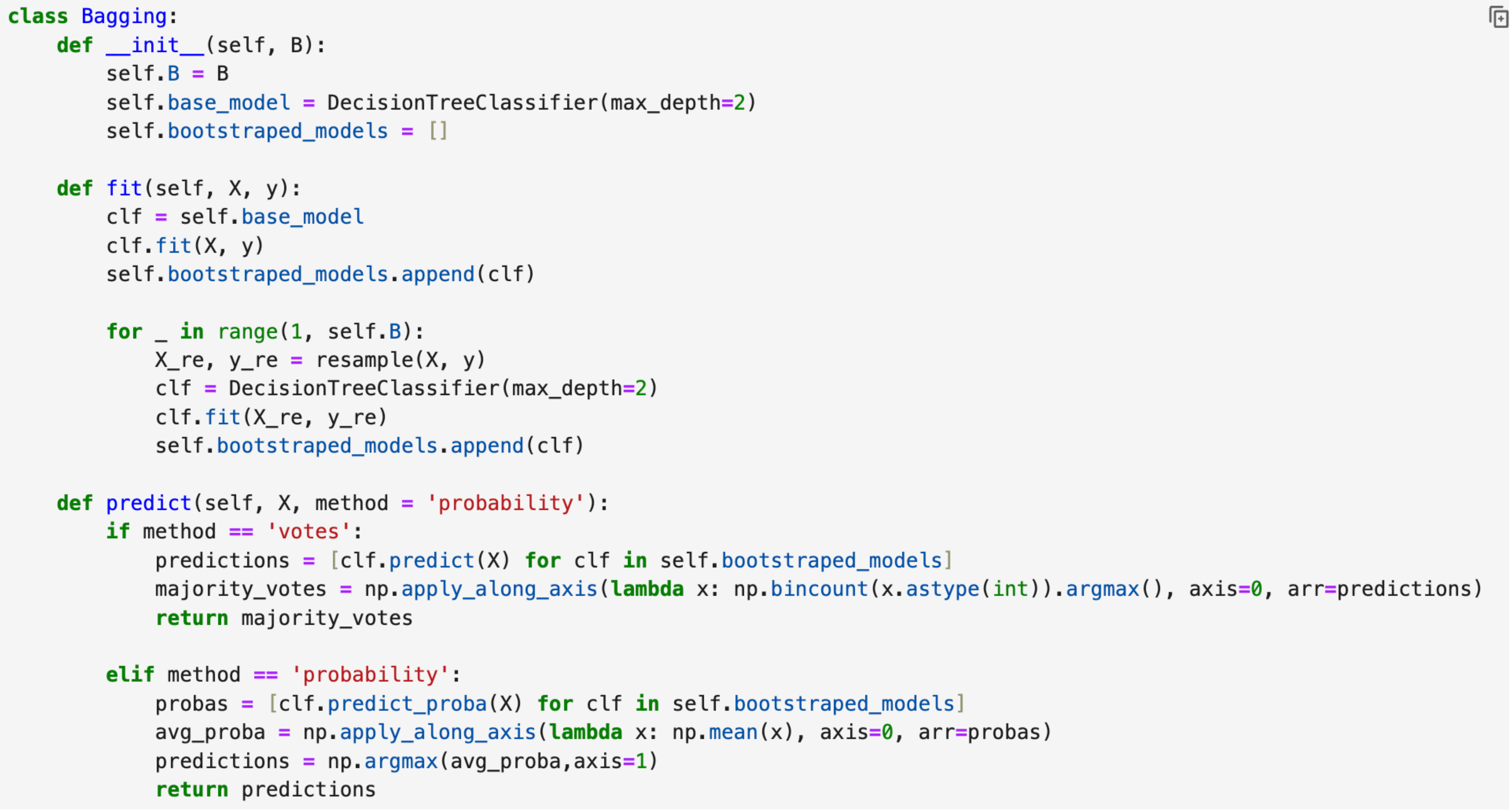


We fit classification trees to the training sample and to each of 200 bootstrap samples.

Notice how the trees are all different, with different splitting features and cut points.

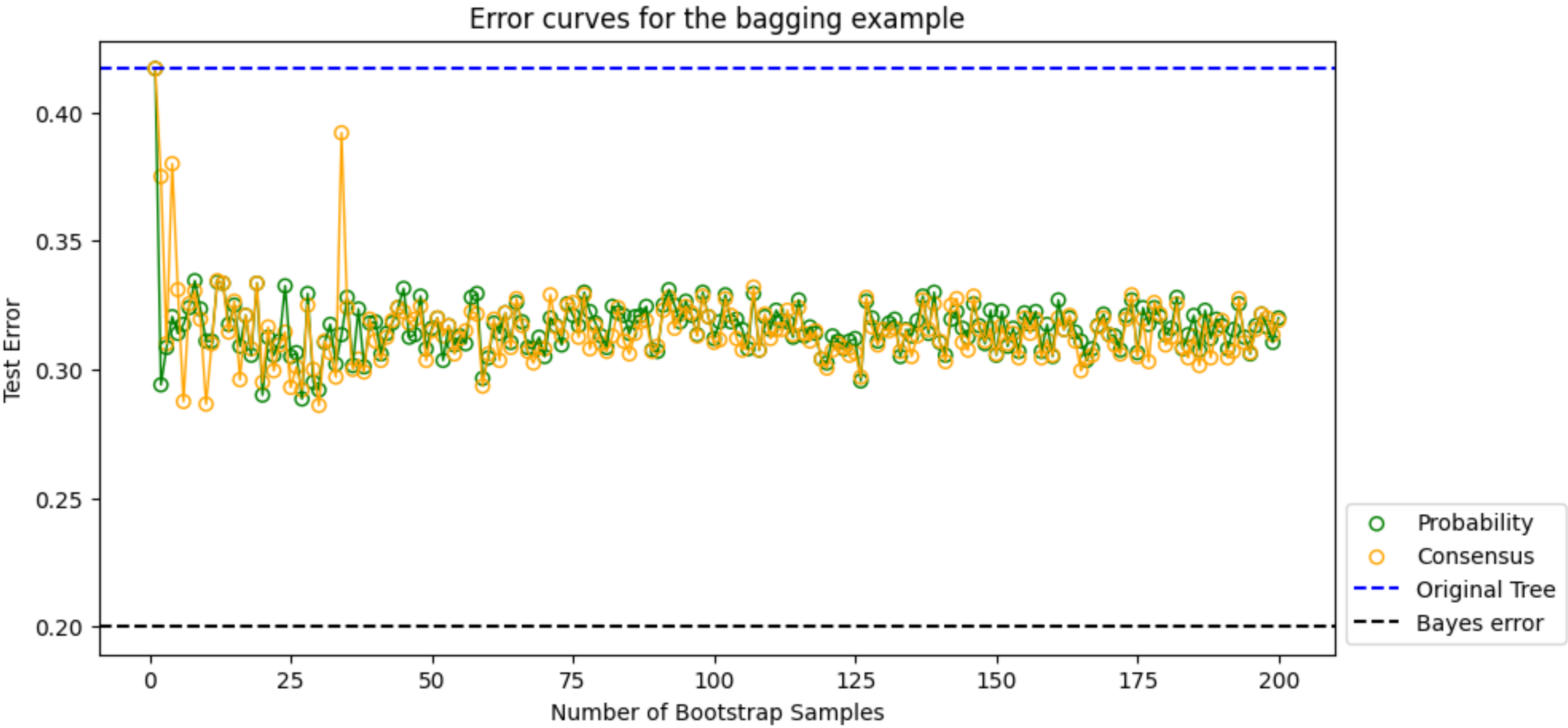
In this example the trees have high variance due to the correlation in the predictors

Bagging



```python
class Bagging:
    def __init__(self, B):
        self.B = B
        self.base_model = DecisionTreeClassifier(max_depth=2)
        self.bootstraped_models = []

    def fit(self, X, y):
        clf = self.base_model
        clf.fit(X, y)
        self.bootstraped_models.append(clf)

        for _ in range(1, self.B):
            X_re, y_re = resample(X, y)
            clf = DecisionTreeClassifier(max_depth=2)
            clf.fit(X_re, y_re)
            self.bootstraped_models.append(clf)

    def predict(self, X, method = 'probability'):
        if method == 'votes':
            predictions = [clf.predict(X) for clf in self.bootstraped_models]
            majority_votes = np.apply_along_axis(lambda x: np.bincount(x.astype(int)).argmax(), axis=0, arr=predictions)
            return majority_votes

        elif method == 'probability':
            probas = [clf.predict_proba(X) for clf in self.bootstraped_models]
            avg_proba = np.apply_along_axis(lambda x: np.mean(x), axis=0, arr=probas)
            predictions = np.argmax(avg_proba,axis=1)
            return predictions
```

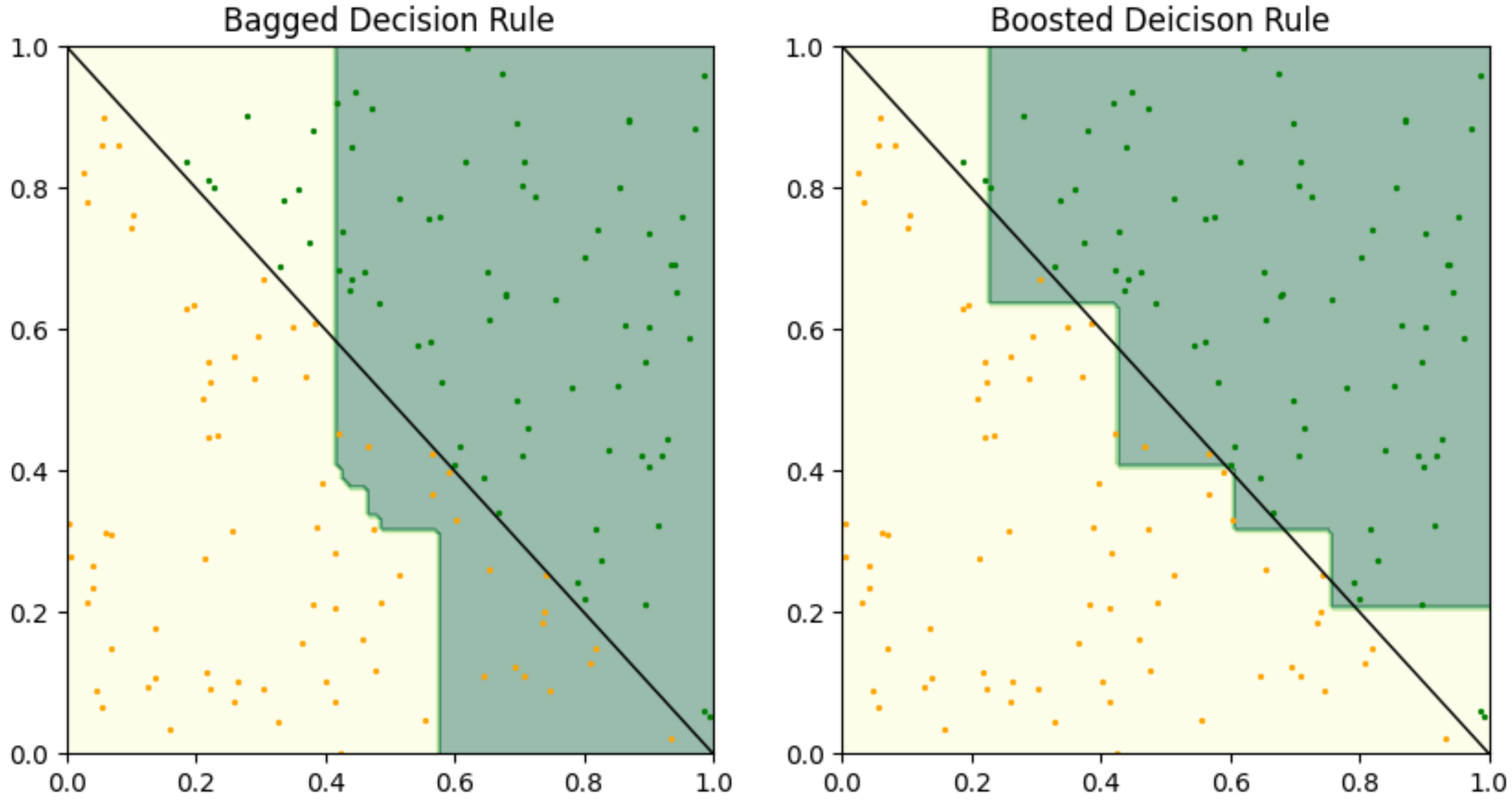
Bagging


Error curves for the bagging example
Test Error
0.40
0.35
0.30
0.25
0.20
0
25
50
75
100
125
150
175
200
Number of Bootstrap Samples
Probability
Consensus
Original Tree
Bayes error

Bagging

- **(Example where bagging doesn't help):** $x = (x_1, x_2)$ **where** $x_i \sim Unif(0,1)$ **and** $y = 1$ **if** $x_1 + x_2 > 1$ **else** 0**.**

- **We choose** $\hat{G}(x)$ **a single axis-oriented split,** $B = 50$**.**



"a woman cannot have 2.4 children"
Averaging over many replications of single split can not captures the diagonal boundary

Bagging doesn't help in this and many other examples where a greater enlargement of the model class is needed

However, boosting roughly captures the diagonal boundary

Model Averaging

- **(Bayesian approach):**

We have a set of candidate models $\mathcal{M}_m, m = 1,..,M$ for our training set $\mathbf{Z}$, and suppose ζ is some quantity of interests (e.g. $f(x_0)$)

$$\mathbb{E}(\zeta|\mathbf{Z}) = \int \zeta Pr(\zeta|\mathbf{Z})d\zeta = \int \sum_{m=1}^{M} \zeta Pr(\zeta|\mathcal{M}_m, \mathbf{Z})Pr(\mathcal{M}_m|\mathbf{Z})d\zeta = \underline{\sum_{m=1}^{M} \mathbb{E}(\zeta|\mathcal{M}_m, \mathbf{Z})Pr(\mathcal{M}_m|\mathbf{Z})}$$

(Committee methods) weights $Pr(\mathcal{M}_m|\mathbf{Z})$ are all equal i.e. unweighted.

Weight average of individual predictions

For the same type of different parameters (e.g. subset in linear regression), BIC criterion can be used to estimate $Pr(\mathcal{M}_m|\mathbf{Z})$

- **(Frequentists approach): linear regression under squared loss example**

Given predictions $\hat{F}(x)^T = (\hat{f}_1(x), \ldots, \hat{f}_M(x))$ with weights $w = (w_1, \ldots, w_M)$ s.t. $\hat{w} = \text{argmin}_w \mathbb{E}\left[Y - \hat{F}(x)^T w\right]^2$

Then, $\mathbb{E}\left[Y - \hat{F}(x)^T w\right]^2 \leq \mathbb{E}\left[Y - \hat{f}_m(x)\right]^2$ $\forall m$. But we can not found $\hat{w}$, and it is natural to replace it with LSE over $\mathbf{Z}$

For the best subset example, it does not work well.. Because we didn't consider the complexity of each model.

Stacking

- Let $\hat{f}_m^{-i}(x)$ be the prediction at x, using model m, applied to the $\mathbf{Z}\backslash\{z_i\}$.
- Then the stacking weights are $\hat{w}^{st} = \text{argmin}_w \sum_{i=1}^{N} \left[y_i - \sum_{m=1}^{M} w_m \hat{f}_m^{-i}(x_i) \right]$ which avoids giving unfairly high weight to models with higher complexity. Better results can be obtained by restricting weights to be non-negative and to sum to 1.
- The stacking estimate is $\sum_m \hat{w}_m^{st} \hat{f}_m(x)$.
- If w is one-hot encoded vector, then this leads to a model choice $\hat{m}$ with smallest LOOCV error.
- Rather than choose single model, stacking combines them with w. This will often lead to better precision, but **less interpretability** than the choice of only one of the M models.