Predictive Uncertainty Estimation with Neural Networks

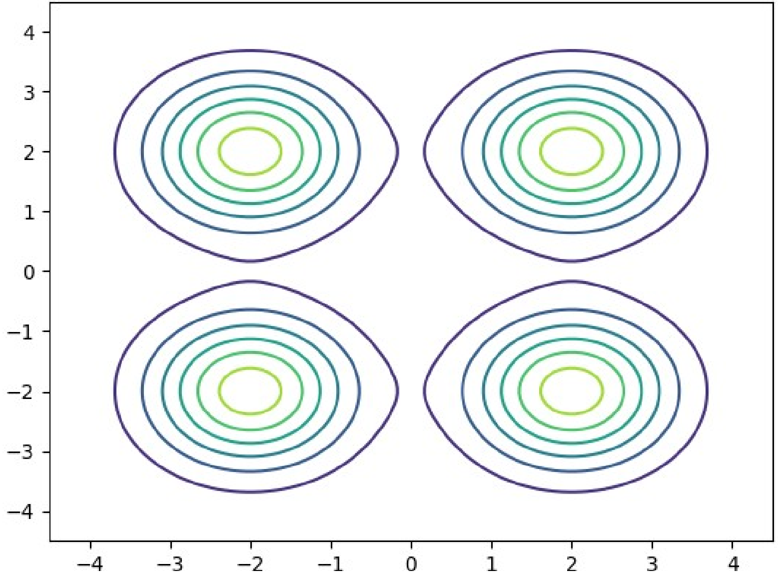
We need to teach how doubt is not to be feared but welcomed. It’s OK to say, “I don’t know.”
- Richard P. Feynman

Types of uncertainty

- In the Bayesian framework, we want our models to capture two different types of uncertainty:
  - **Epistemic (model) uncertainty**: uncertainty in the model parameters.
  - **Aleatoric (data) uncertainty:** inherent and irreducible data noise.

Epistemic (model) uncertainty

- A large set of model parameters explains the data (almost) equally well → large epistemic uncertainty.
- Capturing this uncertainty will help mitigate the problem of **over-confidence** for unseen test inputs.



Aleatoric (data) uncertainty

- **Input-dependent** aleatoric uncertainty is present whenever we expect the estimated targets to be inherently more uncertain for some inputs.

https://youtu.be/lBtRXW9agTQ

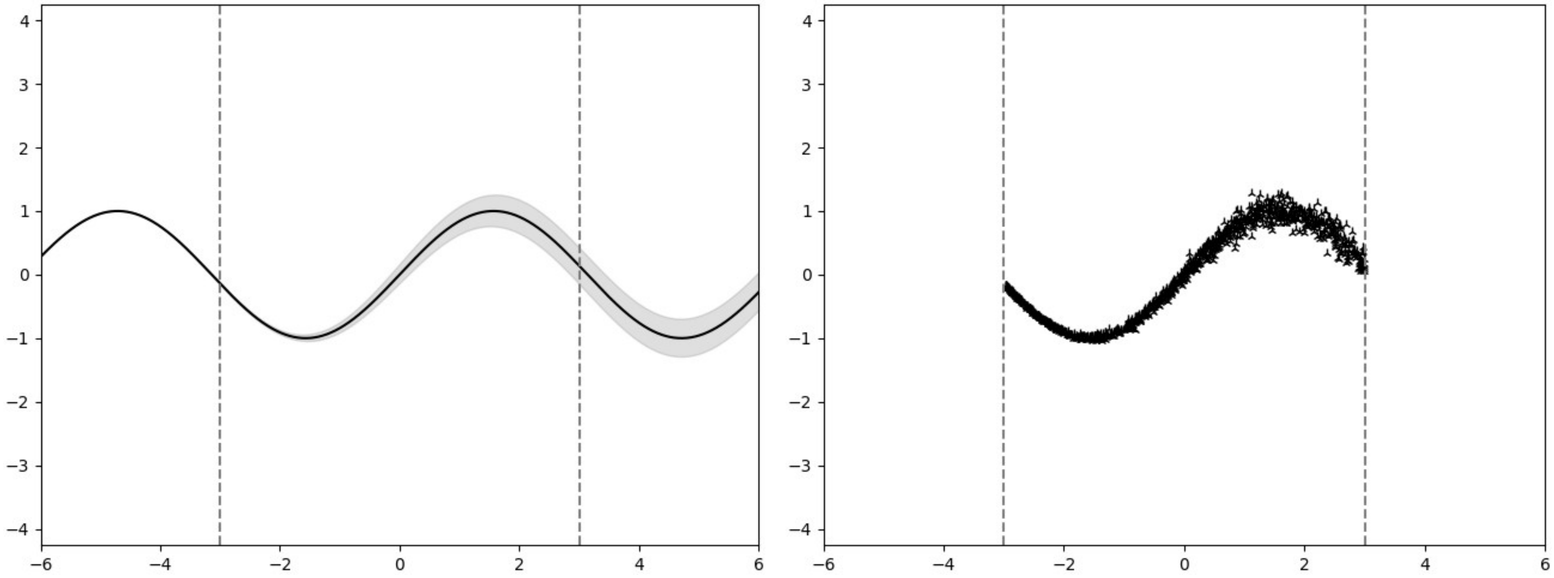
https://youtu.be/KdrHLXpYYlg

Toy regression problem

$$y \sim \mathcal{N}\big(\mu(x), \sigma^2(x)\big), \quad \mu(x) = \sin(x), \quad \sigma(x) = \frac{0.15}{1 + e^{-x}}$$

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}, \quad x_i \sim U[-3, 3]$$

Toy regression problem



Toy regression problem – approach 1

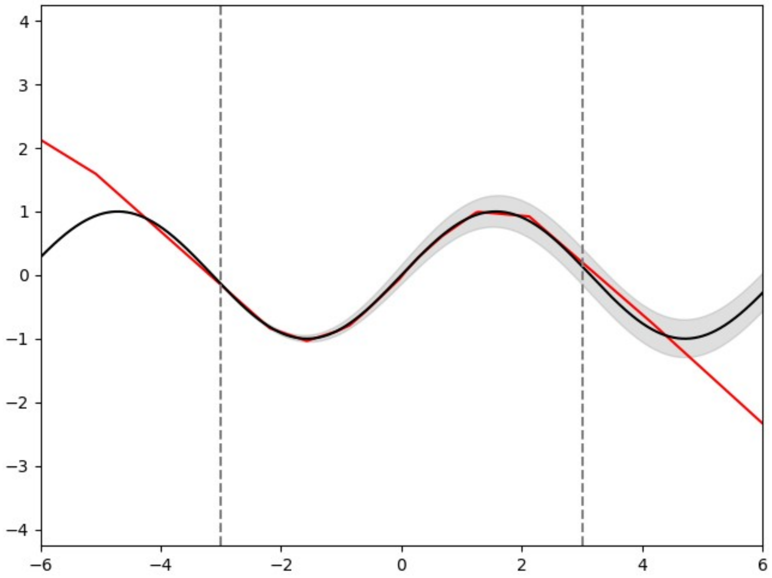
- Directly predict targets using a neural network, find the model parameters by trying to minimize the L2 loss (using SGD):

$$y^* = f_{\hat{\theta}}(x^*)$$

$$\hat{\theta} = \operatorname*{argmin}_{\theta} \frac{1}{N} \sum_{i=1}^{N} (f_\theta(x_i) - y_i)^2$$

Toy regression problem – approach 1

- We are able to match the true mean almost perfectly in the training data interval.
- Our model does however fail to capture **both** epistemic uncertainty and the input-dependent aleatoric uncertainty.



Toy regression problem – approach 2

- Explicitly model the conditional distribution using a neural network, try to find the MLE of the model parameters (using SGD):

$$p(y|x,\theta) = \mathcal{N}\big(\mu_\theta(x),\, \sigma^2_\theta(x)\big), \quad f_\theta(x) = [\,\mu_\theta(x) \quad \log\sigma^2_\theta(x)\,]^T \in \mathbb{R}^2$$

$$p(\mathcal{D}|\theta) = \prod_{i=1}^{N} p(y_i|x_i,\theta)$$

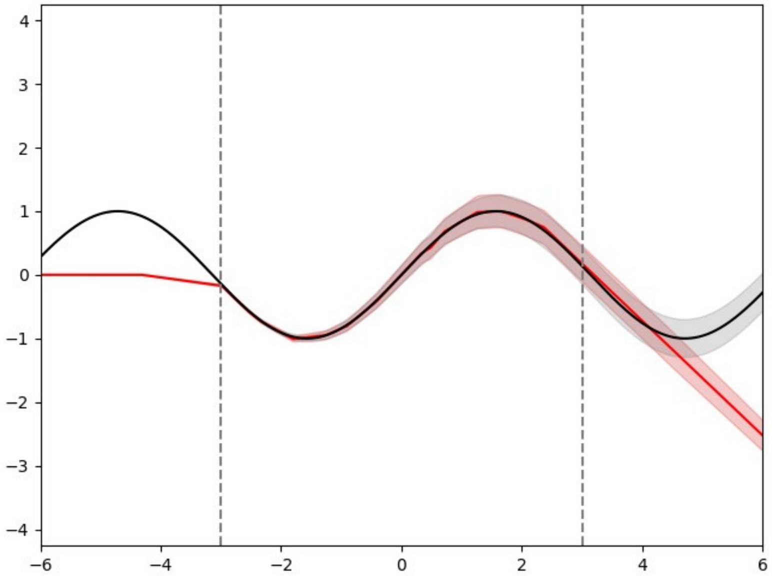
$$\hat{\theta}_{\text{MLE}} = \underset{\theta}{\text{argmin}}(-\log p(\mathcal{D}|\theta))$$

- Obtained predictive distribution:

$$p(y^*|x^*,\hat{\theta}_{\text{MLE}}) = \mathcal{N}\big(\mu_{\hat{\theta}_{\text{MLE}}}(x^*),\, \sigma^2_{\hat{\theta}_{\text{MLE}}}(x^*)\big)$$

Toy regression problem – approach 2

- Our predictive distribution closely matches the true conditional distribution in the training data interval → captures aleatoric uncertainty.
- Our model does however still become highly over-confident outside this interval since it fails to capture **epistemic** uncertainty.



Toy regression problem – approach 3

- Instead of MLE, we employ Bayesian inference to obtain a different predictive distribution:

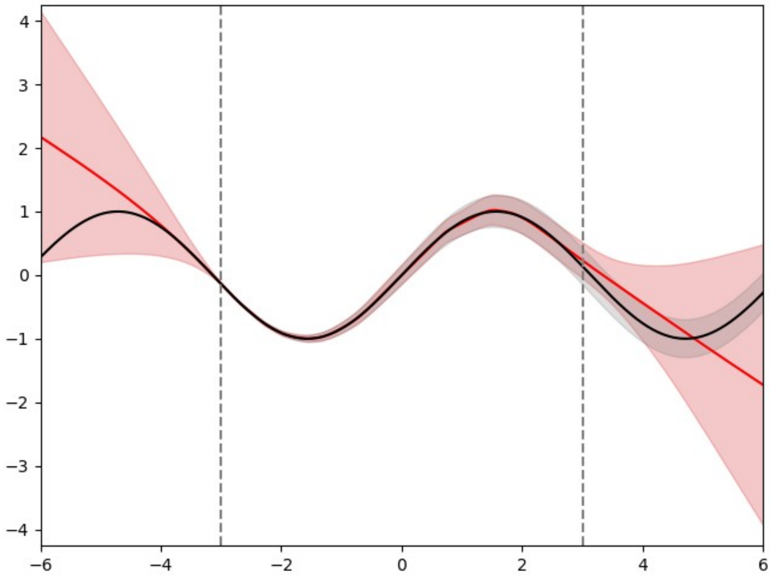
$$p(y^*|x^*, \mathcal{D}) = \int p(y^*|x^*, \theta) p(\theta|\mathcal{D}) d\theta \approx \frac{1}{M} \sum_{i=1}^{M} p(y^*|x^*, \theta^{(i)}), \theta^{(i)} \sim p(\theta|\mathcal{D})$$

- Implemented using M = 1000 samples obtained via Hamiltonian Monte Carlo (using Pyro).
- (we also approximate the uniformly weighted mixture of Gaussians one obtains with a single Gaussian)

Toy regression problem – approach 3

- The estimated uncertainty now also increases appropriately as the estimated mean diverges from the true value outside the training data interval
→ the model does **not** become over-confident.

- Our model is thus able to estimate both aleatoric and epistemic uncertainty.



Toy regression problem – approach 4

- Unfortunately, Hamiltonian Monte Carlo (and similar MCMC methods) is not scalable to large models and/or datasets.

- Instead, we view **ensembling** as approximate Bayesian inference to obtain another predictive distribution:

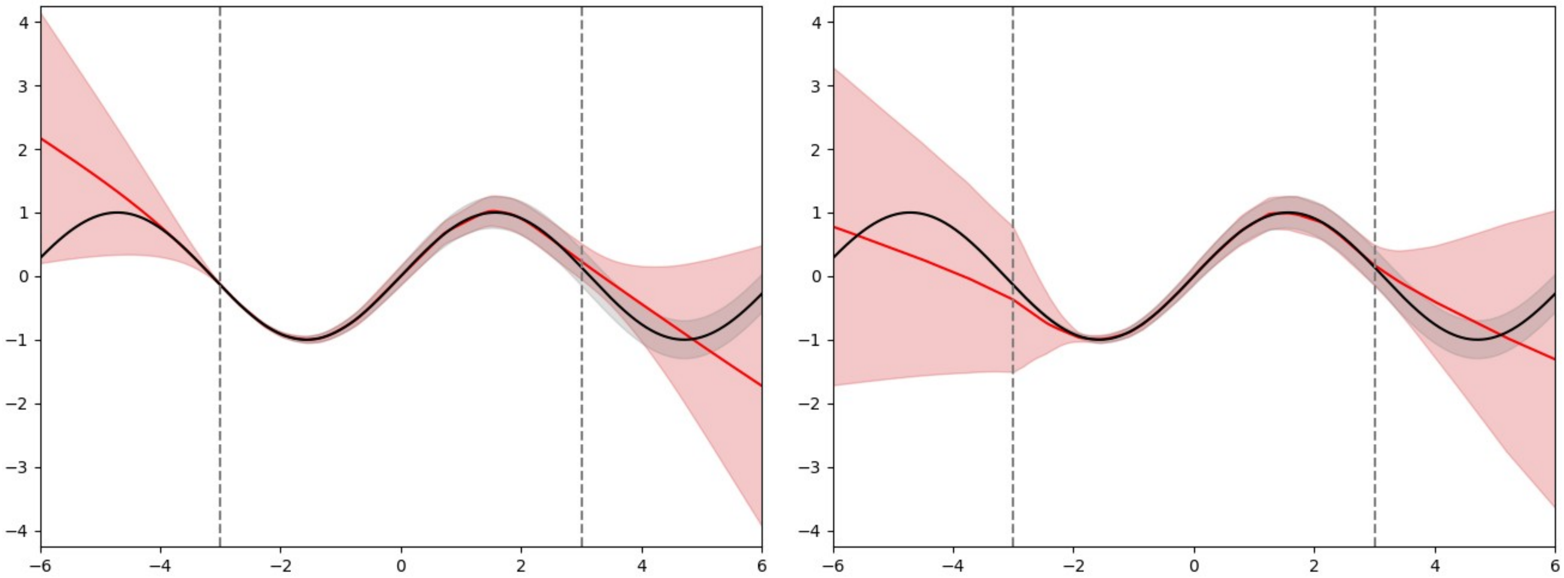
$$\hat{p}(y^*|x^*, \mathcal{D}) = \frac{1}{M}\sum_{i=1}^{M} p(y^*|x^*, \theta^{(i)})$$

- Where $\{\theta^{(1)}, \ldots, \theta^{(M)}\}$ is obtained by independently training M models with **random initialization**, by trying to minimize the negative log-likelihood (as in approach 2).

- (we can also add a prior for θ and try to minimize the MAP objective instead)

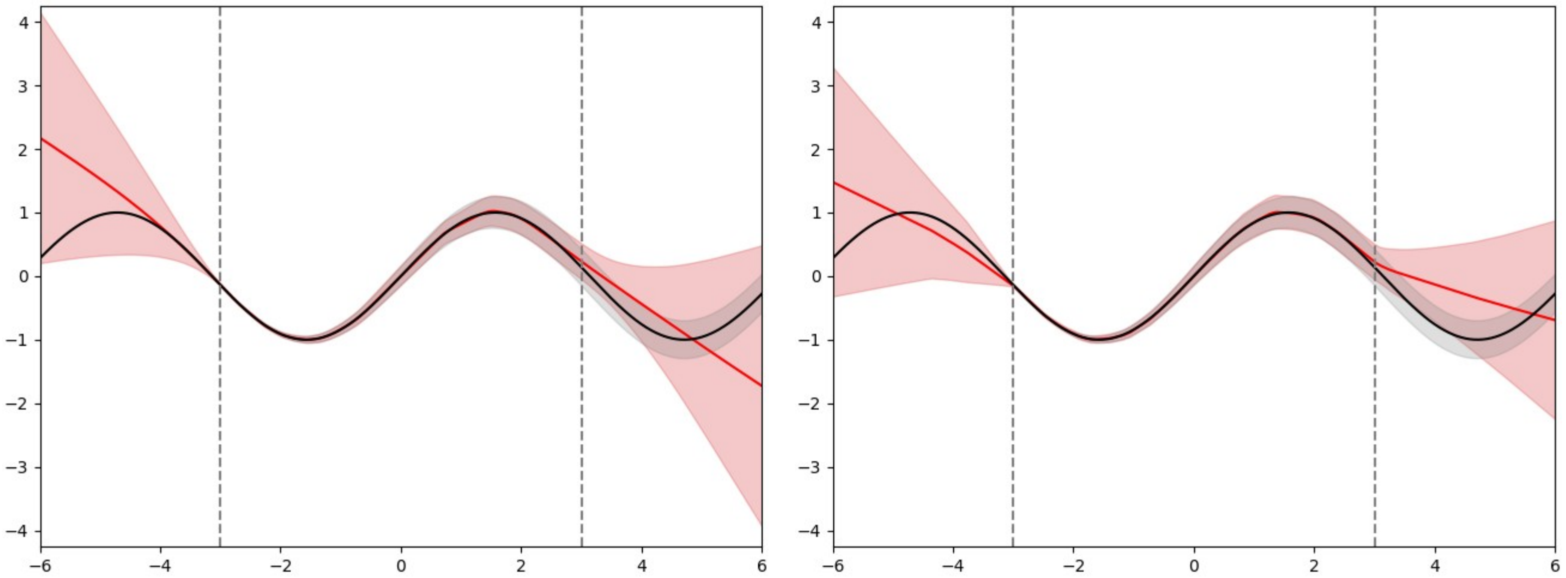
Toy regression problem – approach 4

- M = 4.



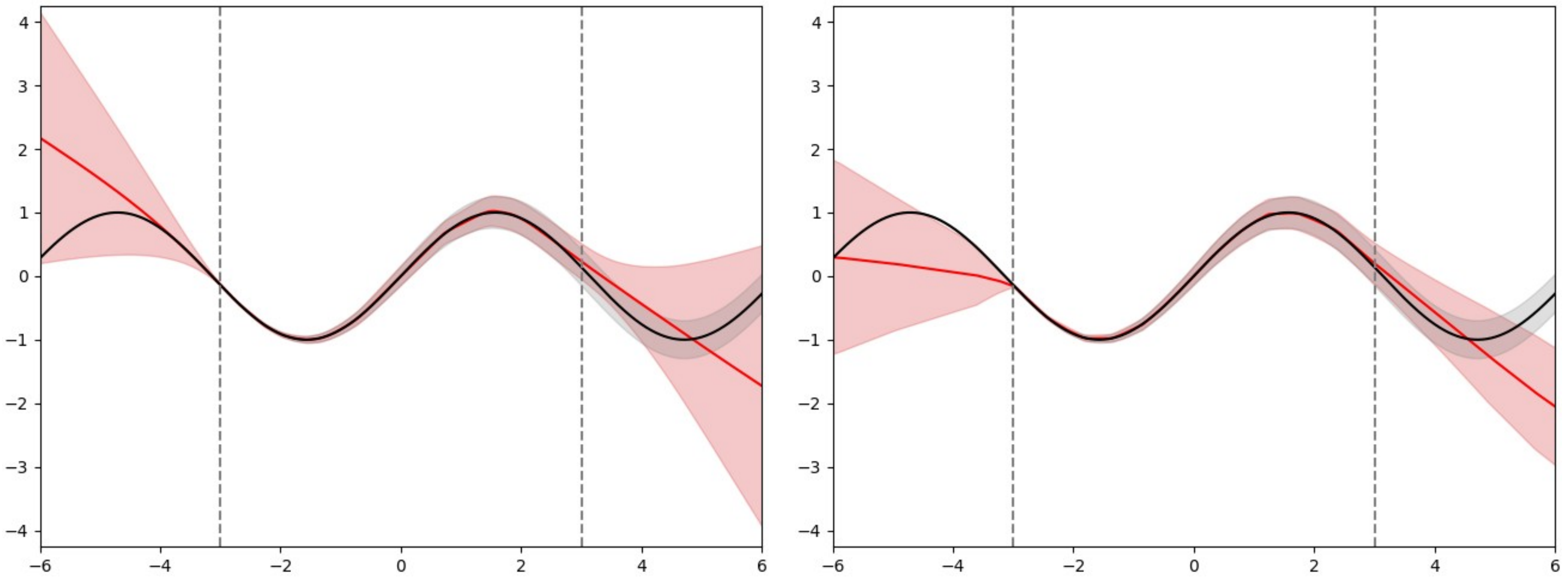
Toy regression problem – approach 4

- M = 4.



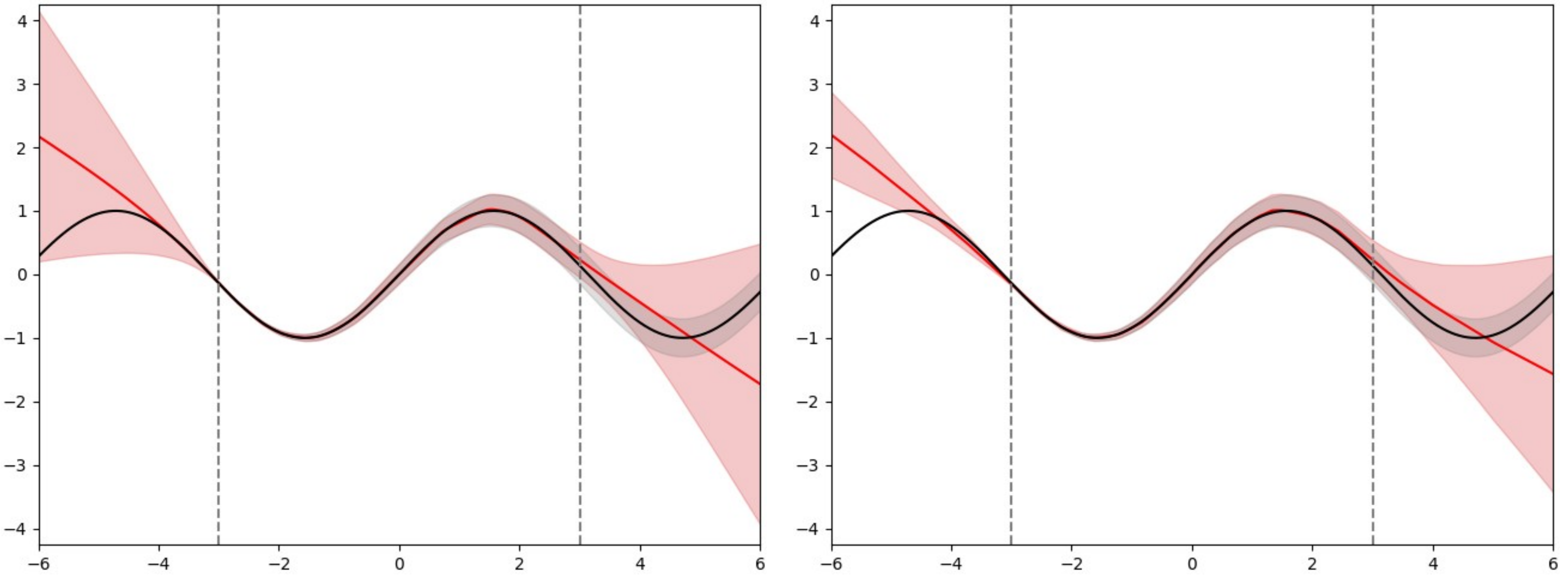
Toy regression problem – approach 4

- M = 4.



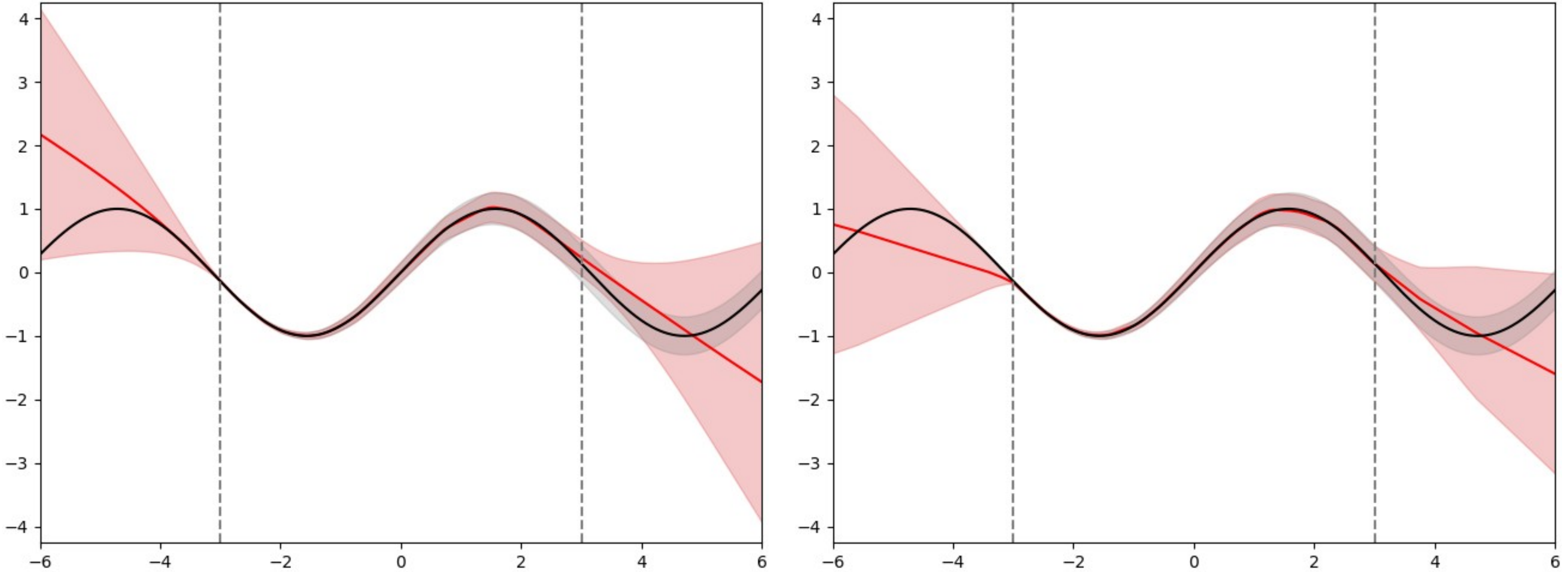
Toy regression problem – approach 4

- M = 4.



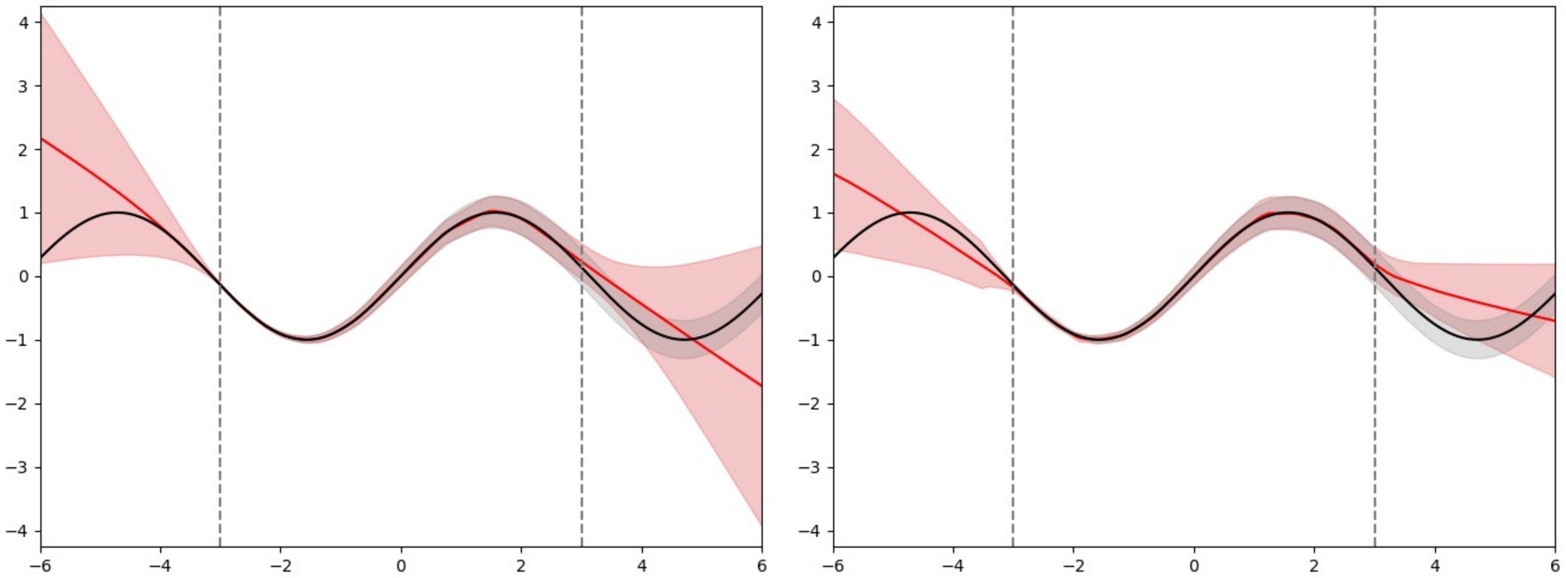
Toy regression problem – approach 4

- M = 4.



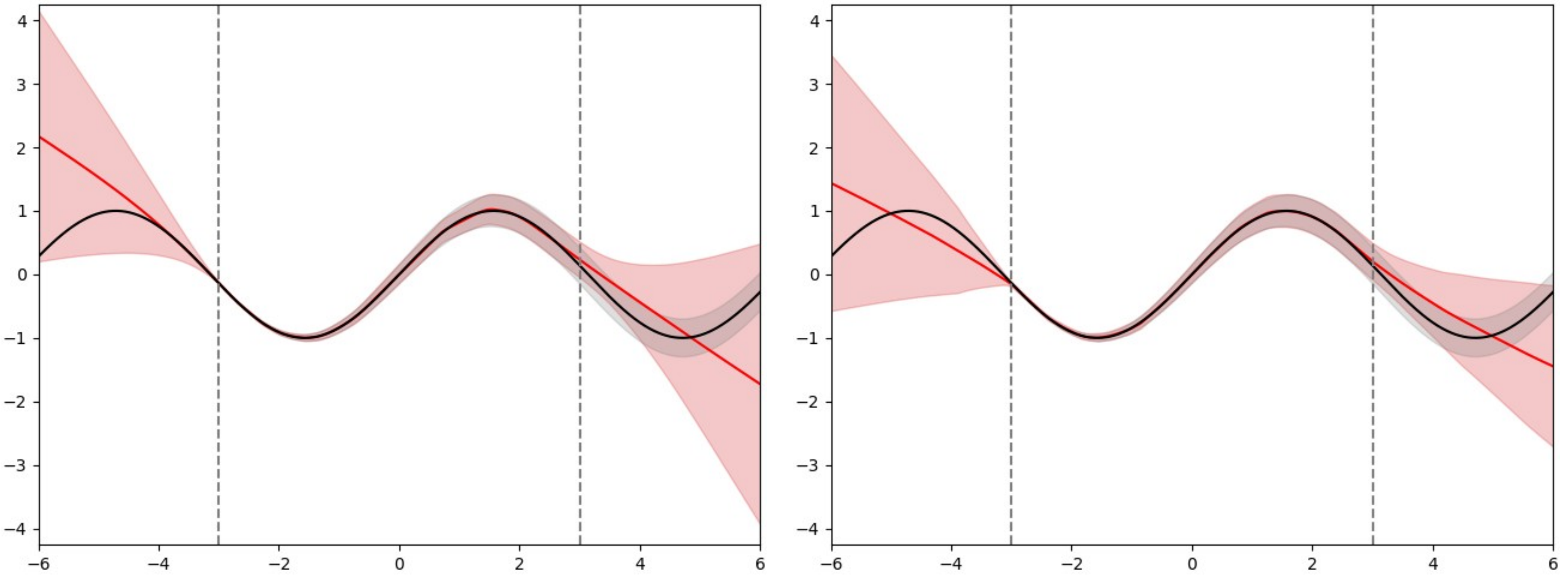
Toy regression problem – approach 4

- M = 4.



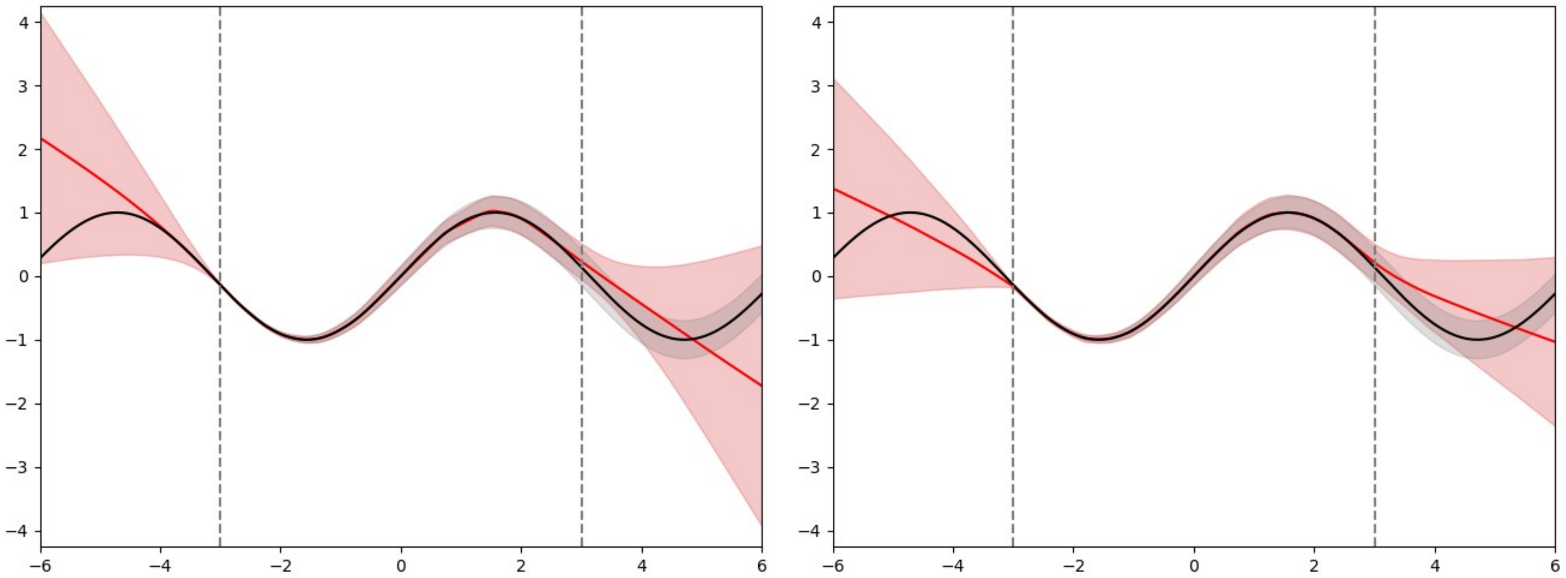
Toy regression problem – approach 4

- M = 16.



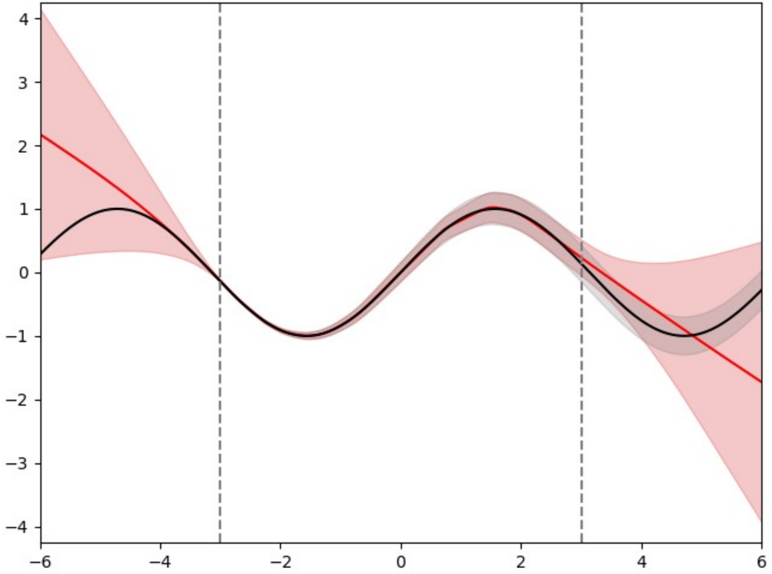
Toy regression problem – approach 4

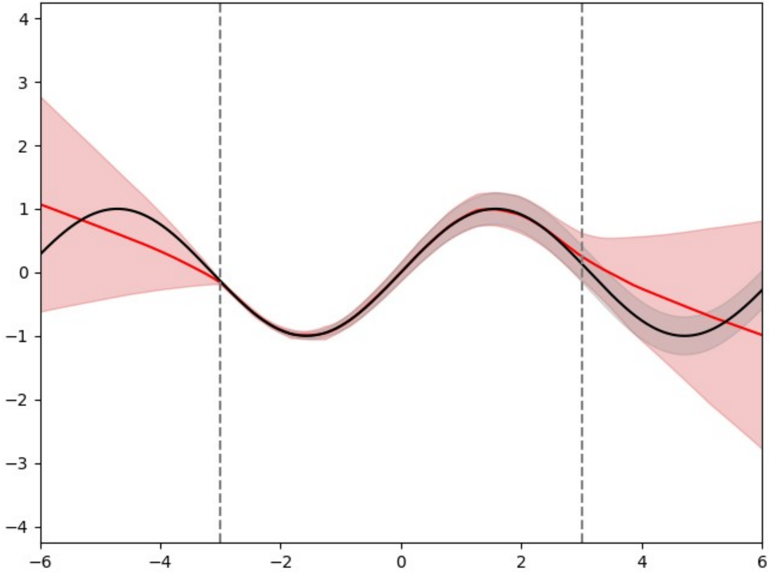
- M = 16.



Toy regression problem – approach 4

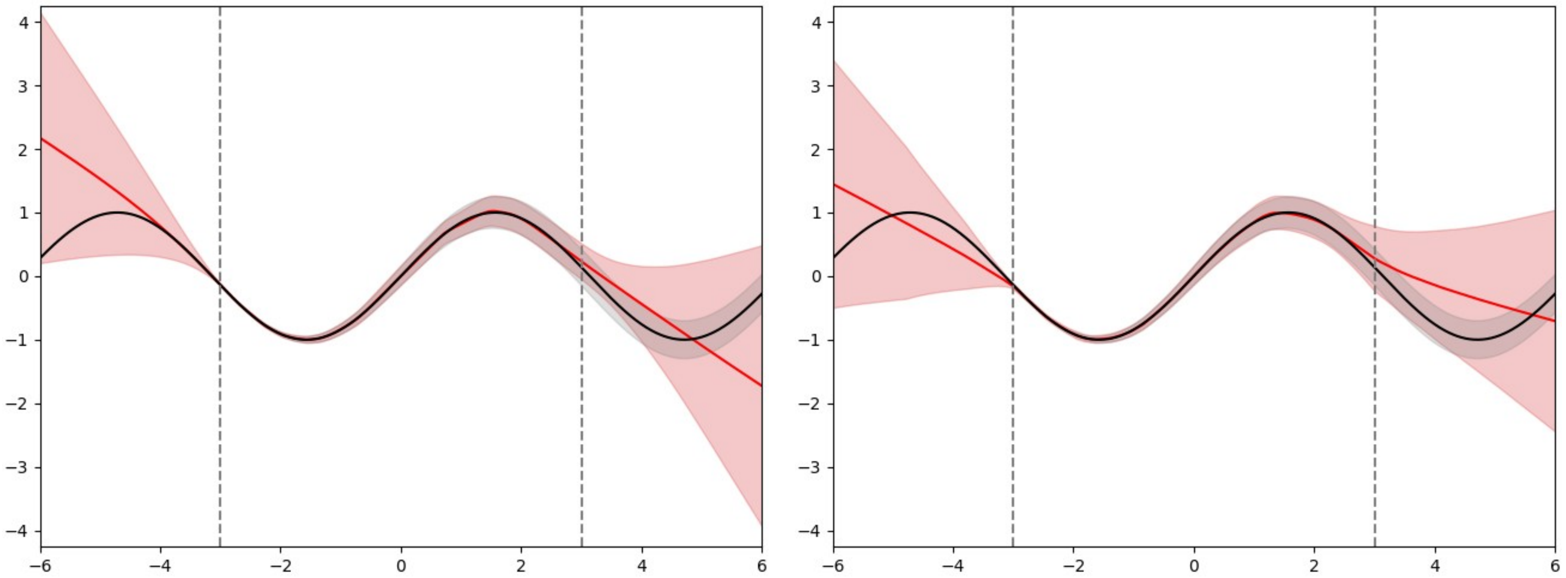
- M = 16.





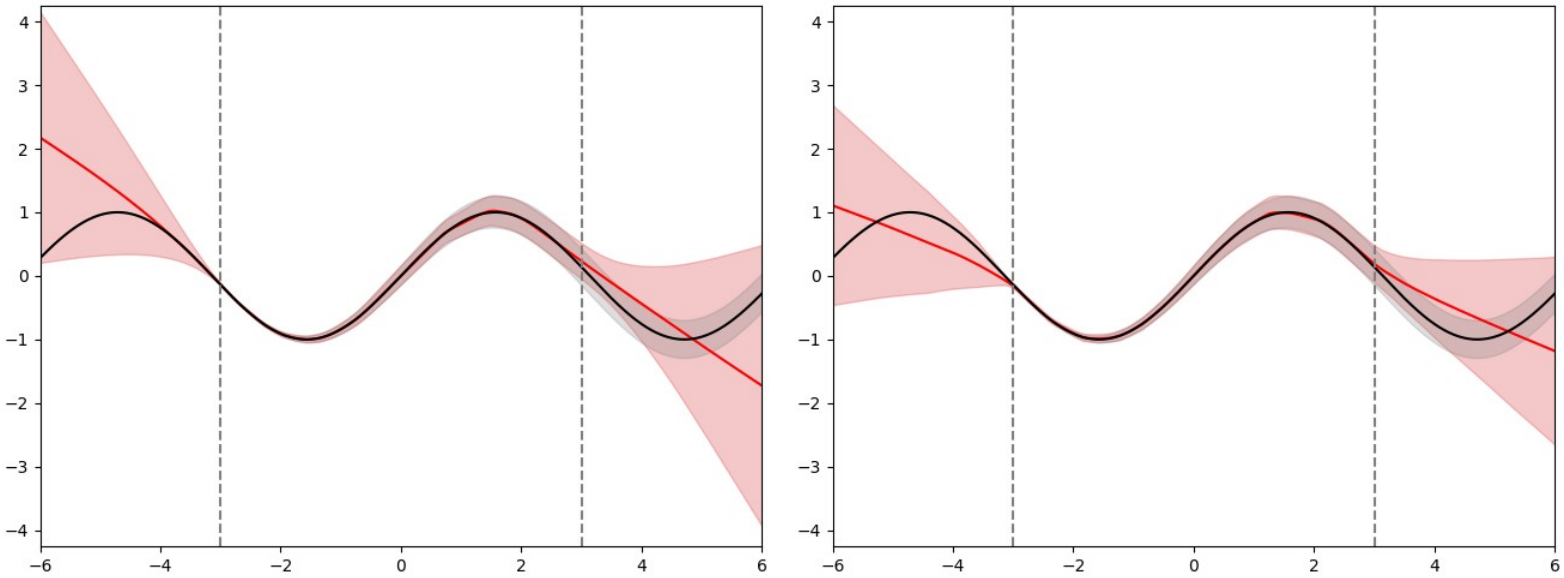
Toy regression problem – approach 4

- M = 16.



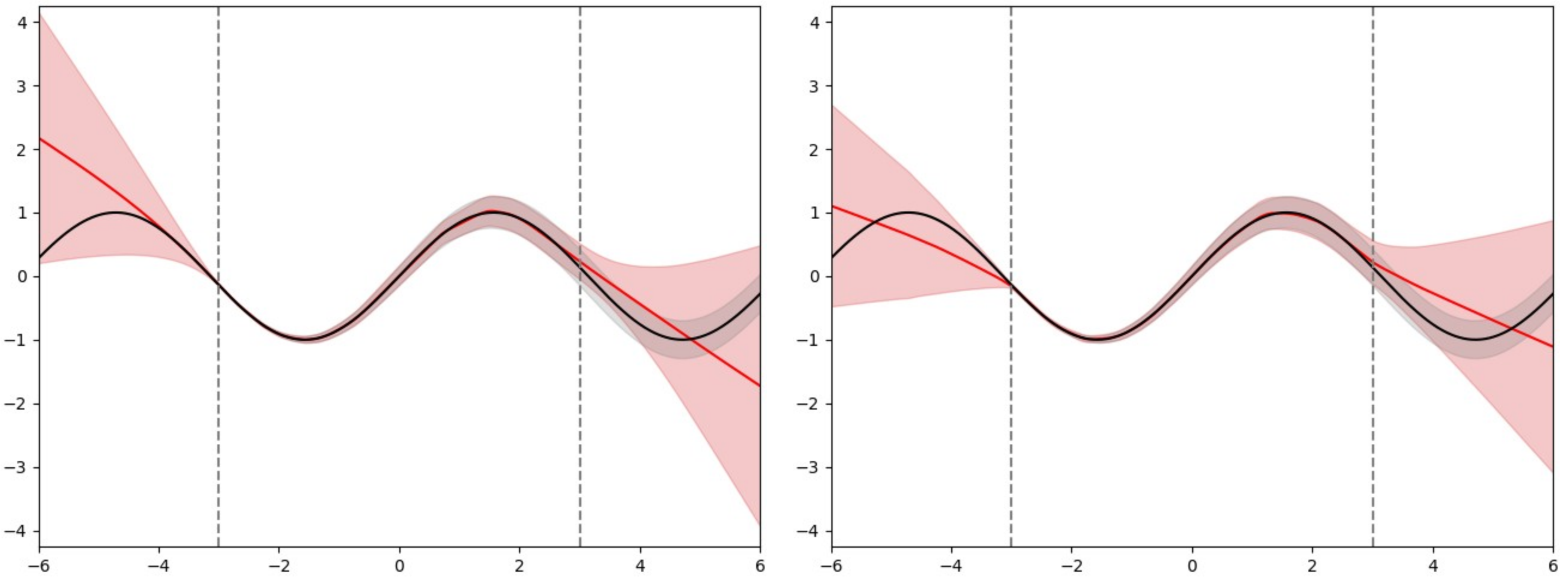
Toy regression problem – approach 4

- M = 16.



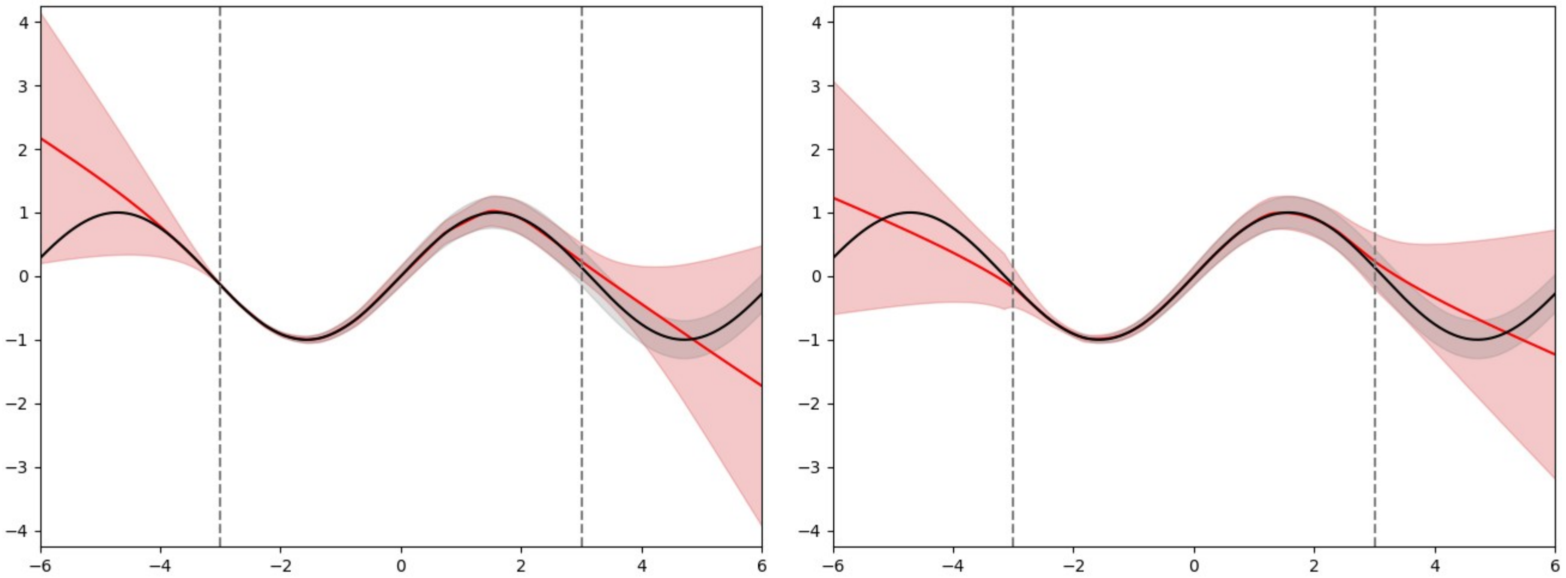
Toy regression problem – approach 4

- M = 16.



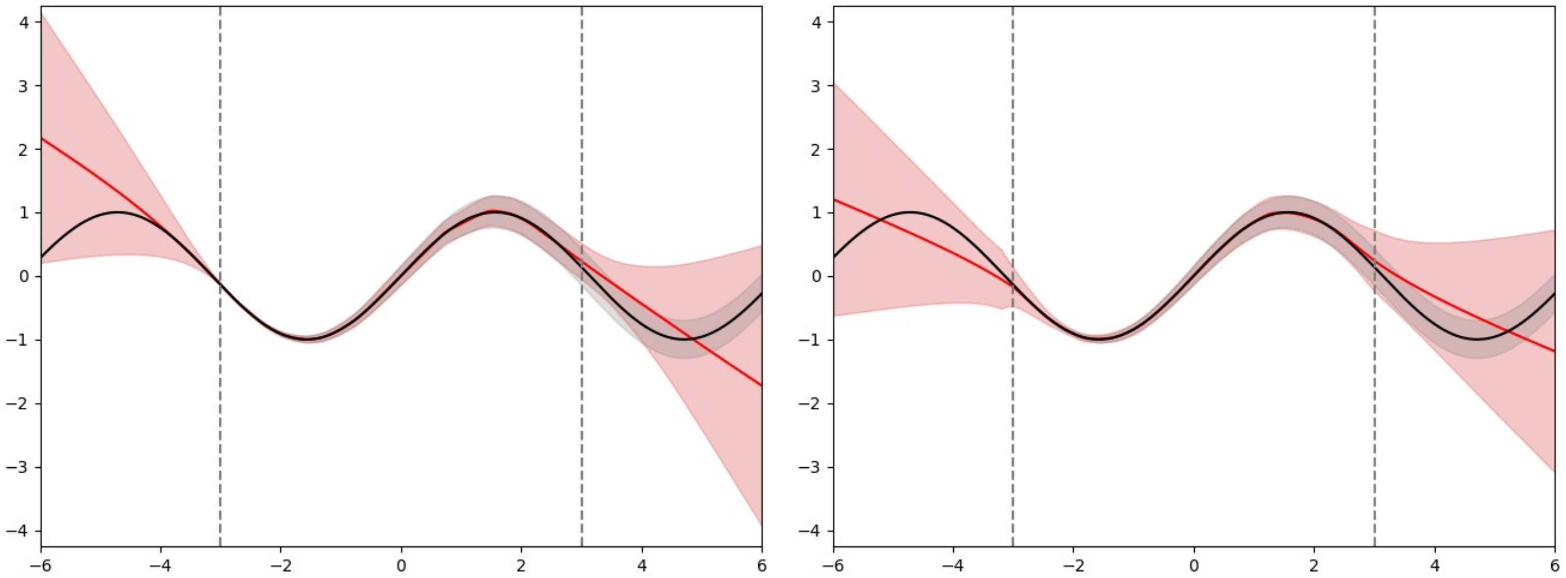
Toy regression problem – approach 4

- M = 256.



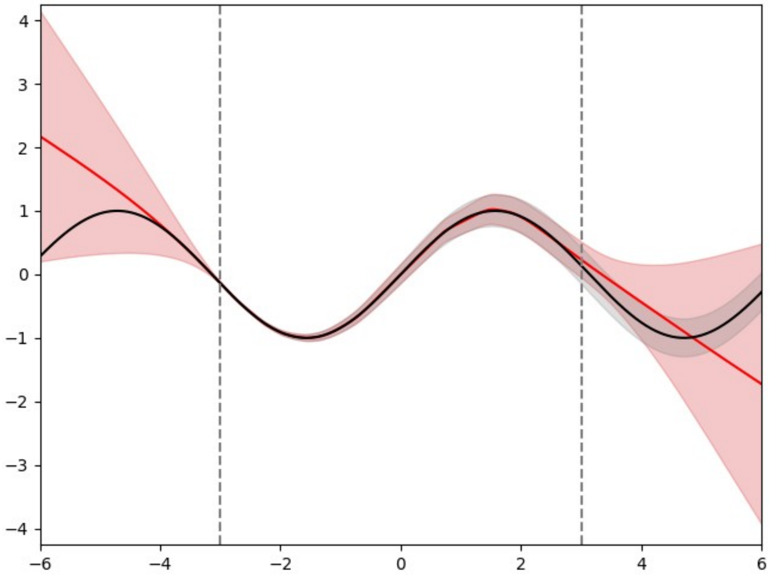
Toy regression problem – approach 4

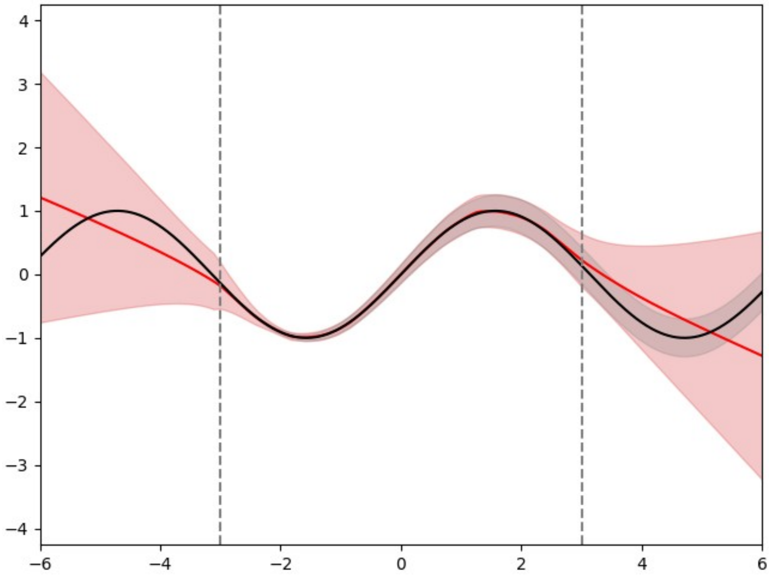
- M = 256.



Toy regression problem – approach 4

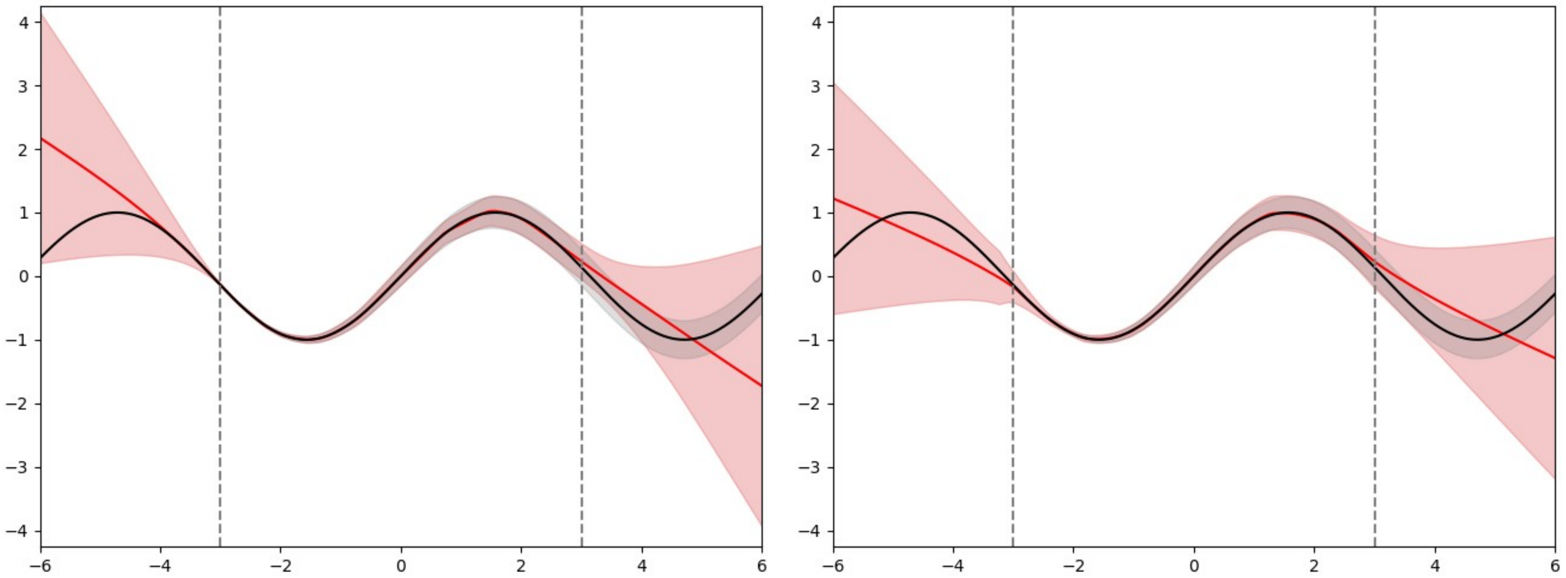
- M = 256.





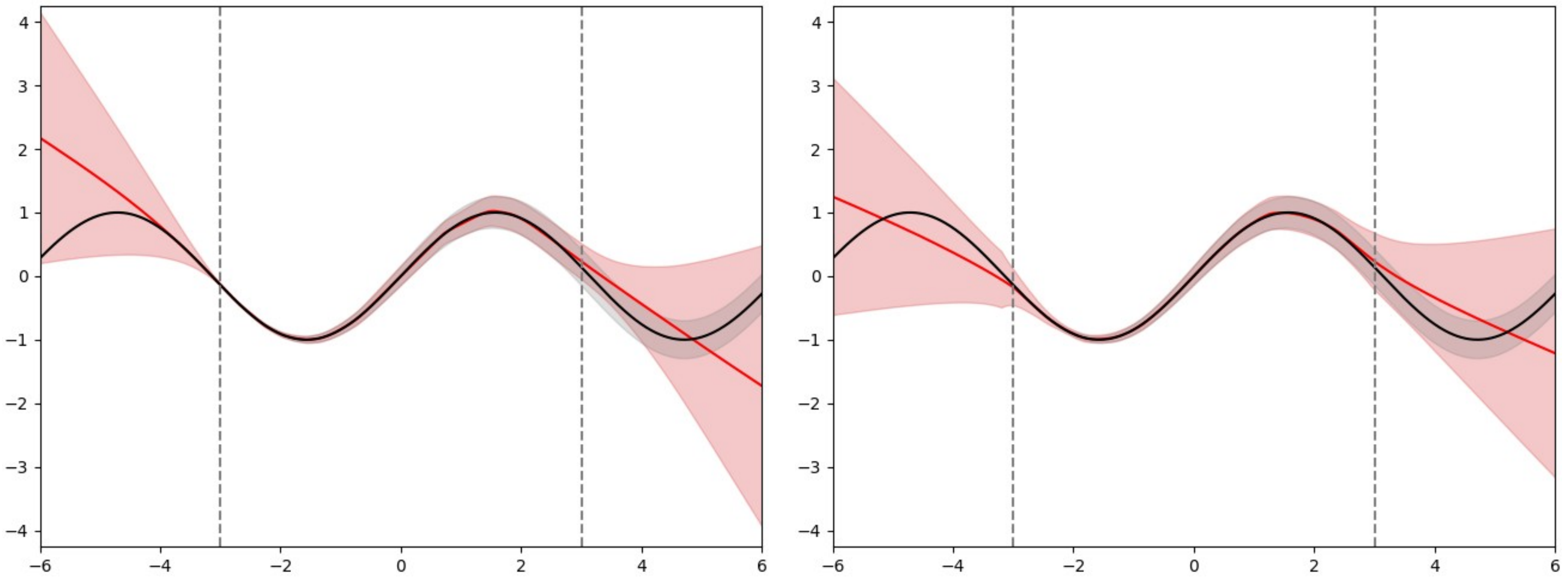
Toy regression problem – approach 4

- M = 256.



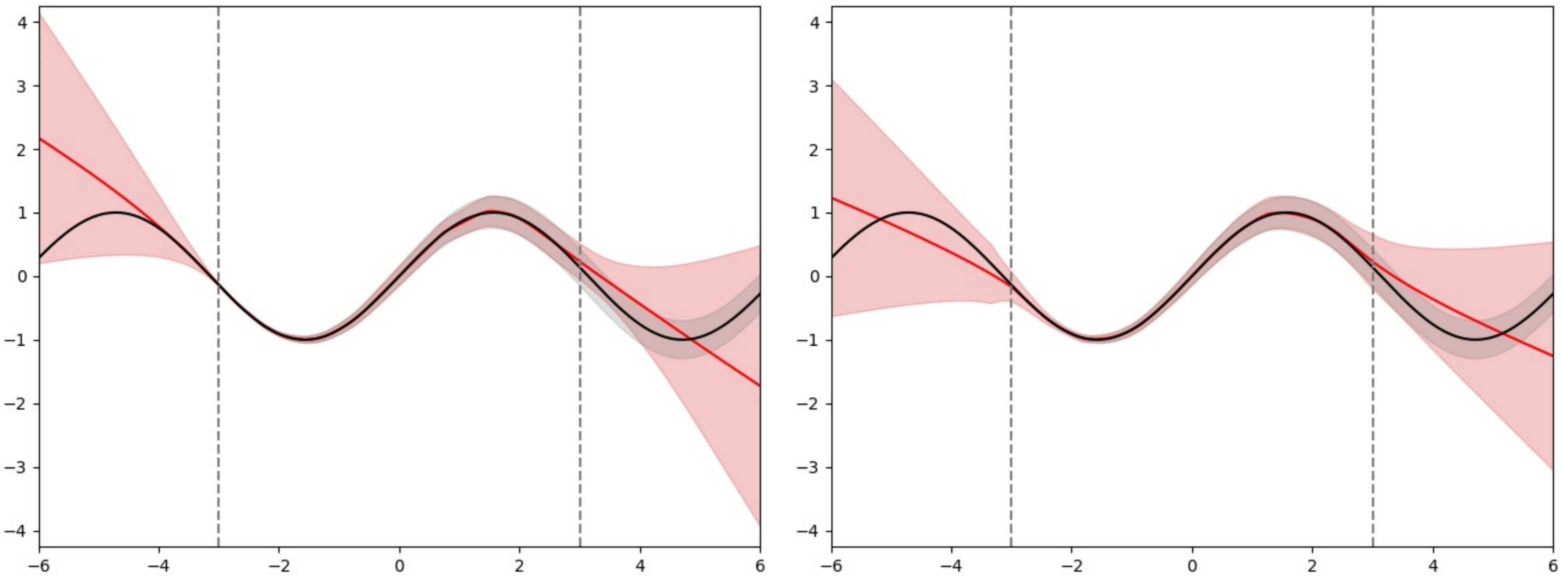
Toy regression problem – approach 4

- M = 256.



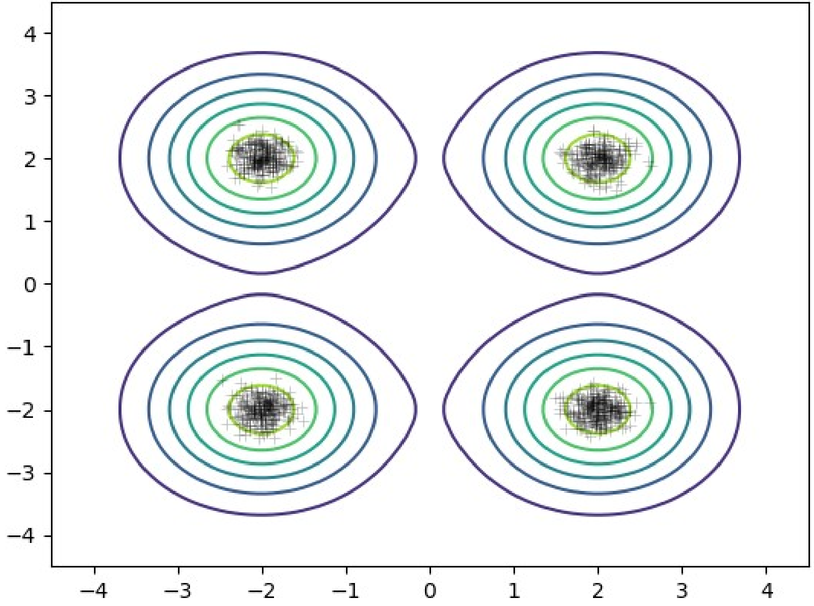
Toy regression problem – approach 4

- M = 256.



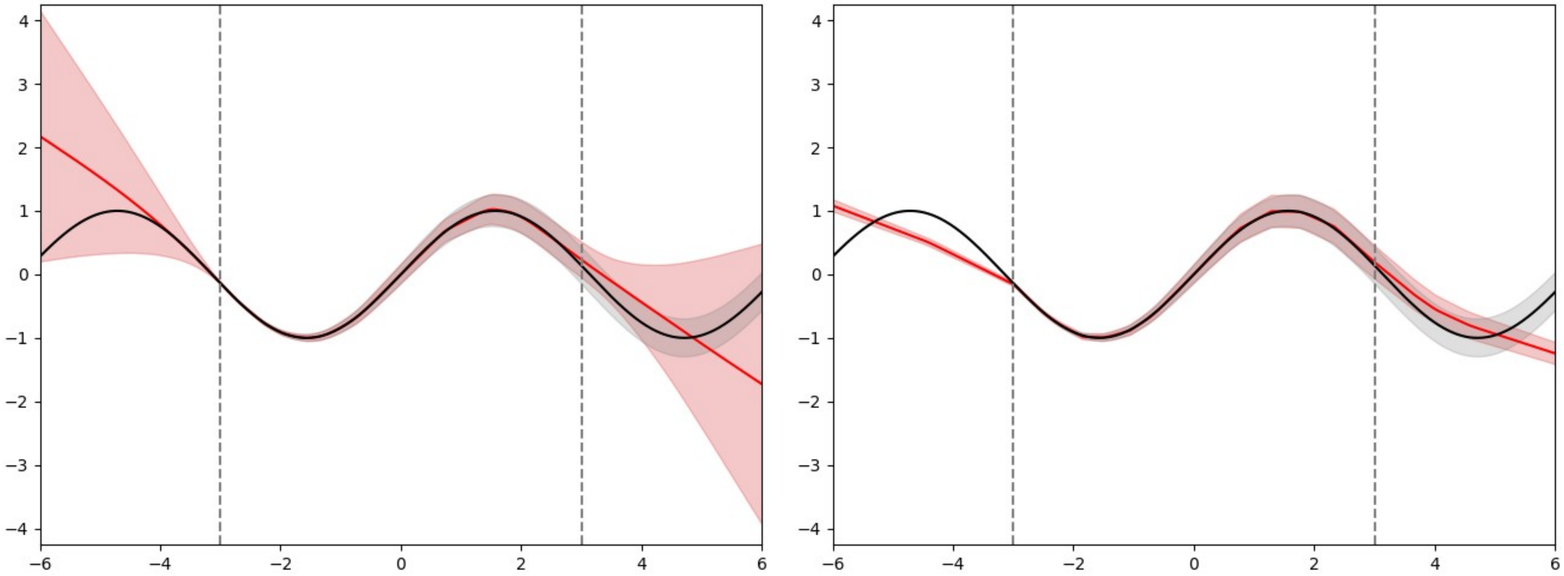
Toy regression problem – approach 4

- Why does this approach even work at all?
- Probably because the data likelihood / posterior is highly **multi-modal** for neural networks. Most local maxima also seem to have similar maximum values.
- When we try to minimize the corresponding loss multiple times (using SGD), starting from **random initial points**, we will thus likely end up at different local modes, capturing some of this multi-modality.



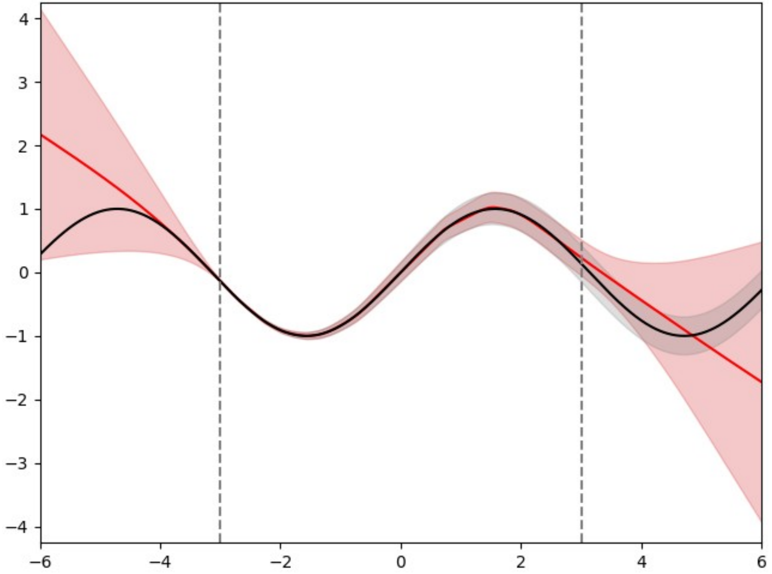
Toy regression problem – approach 4

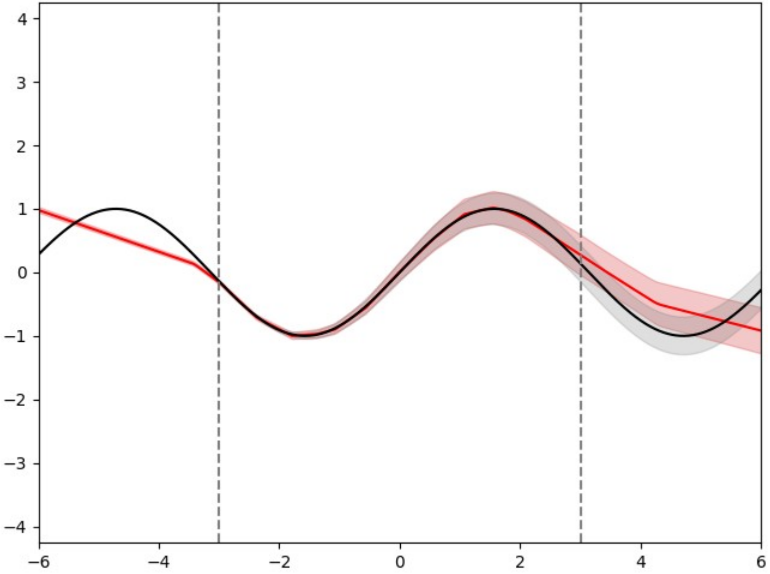
- M = 64, but all networks are trained with the **same** (randomly chosen) initialization.



Toy regression problem – approach 4

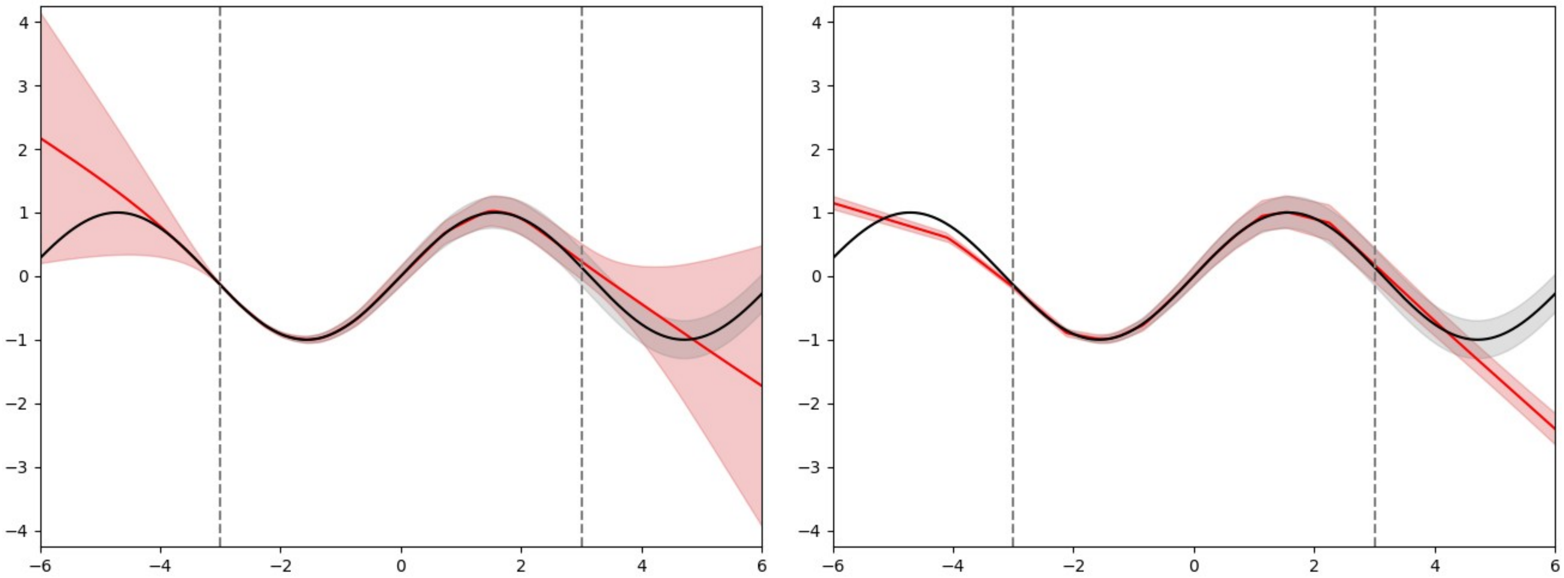
- M = 64, but all networks are trained with the **same** (randomly chosen) initialization.



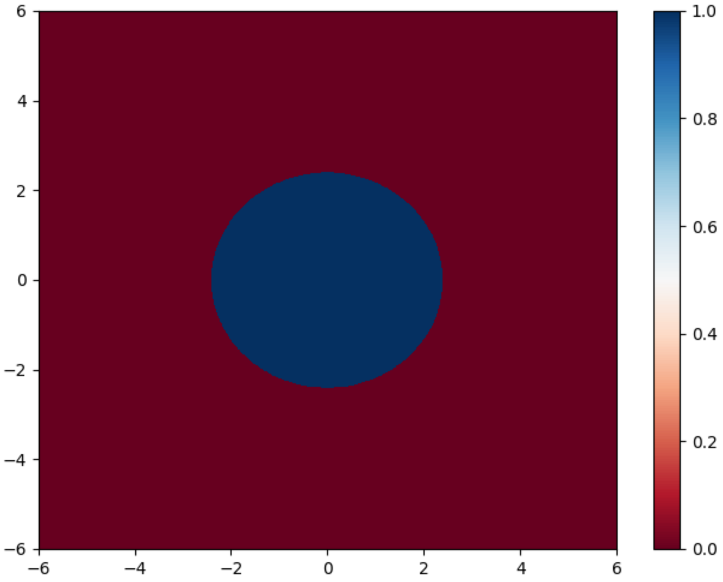


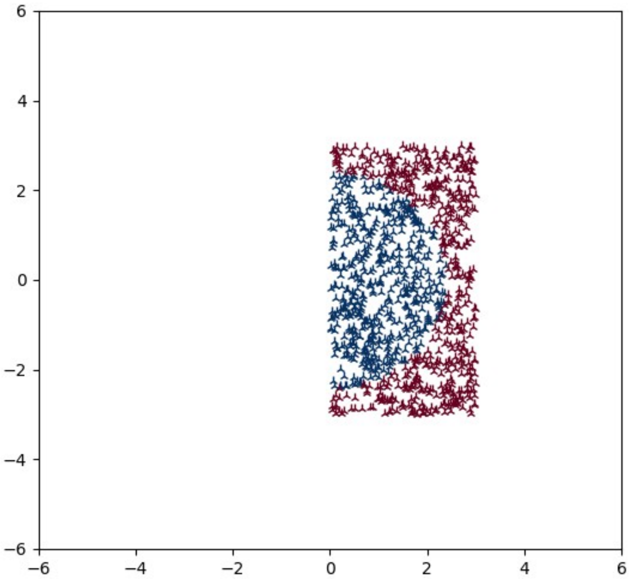
Toy regression problem – approach 4

- M = 64, but all networks are trained with the **same** (randomly chosen) initialization.



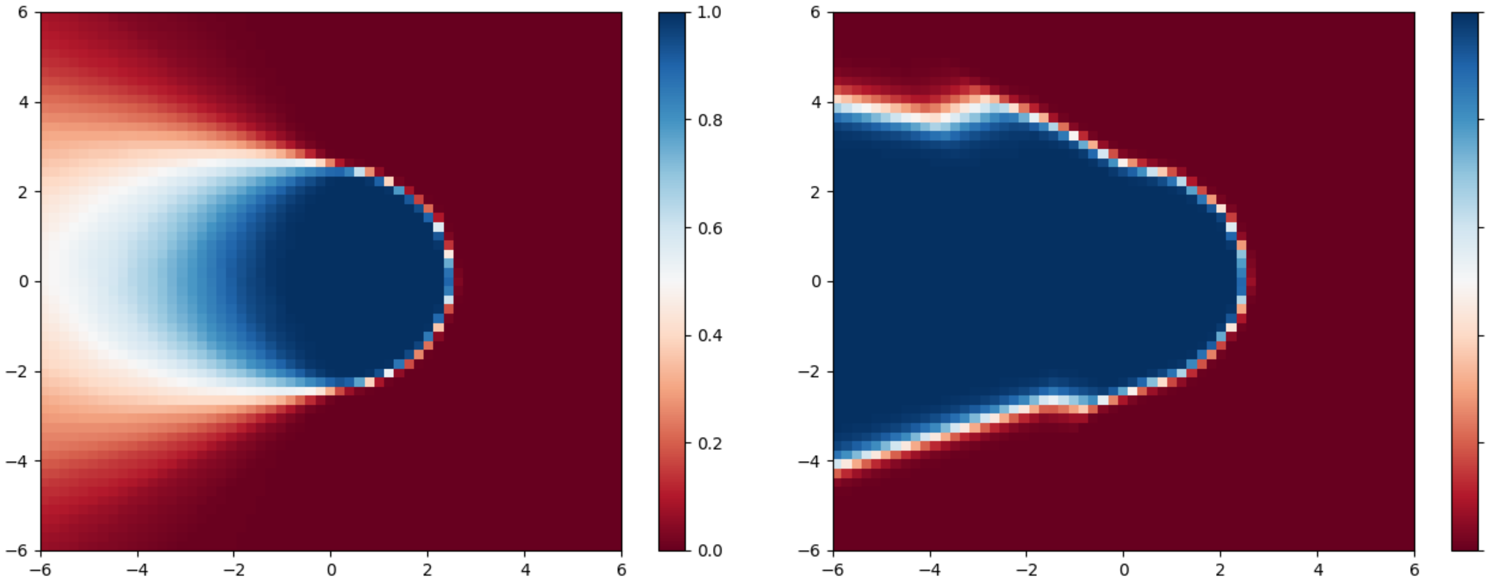
Toy classification problem





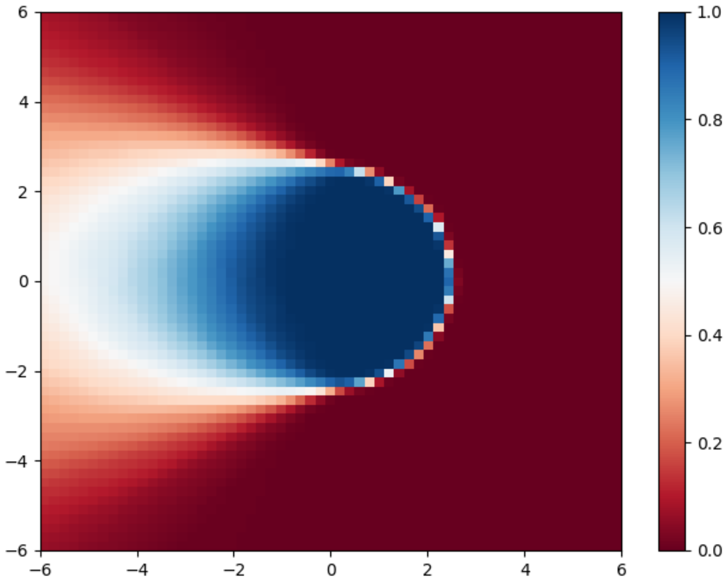
Toy classification problem – approach 4

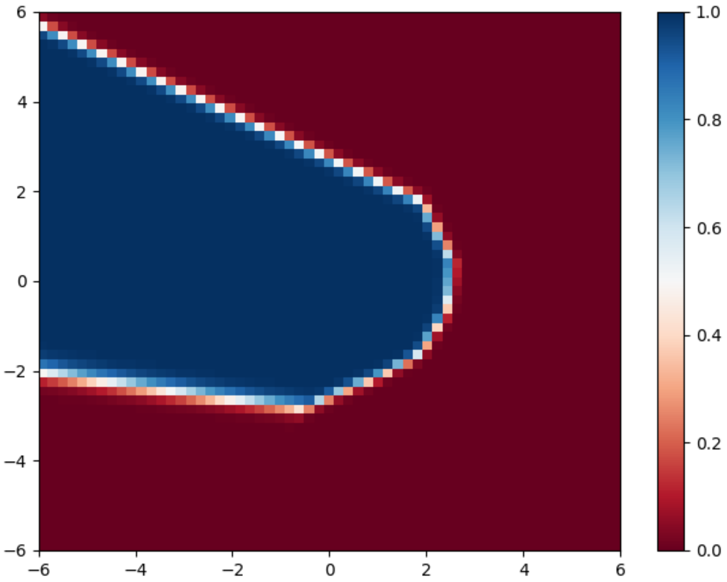
- M = 1.



Toy classification problem – approach 4

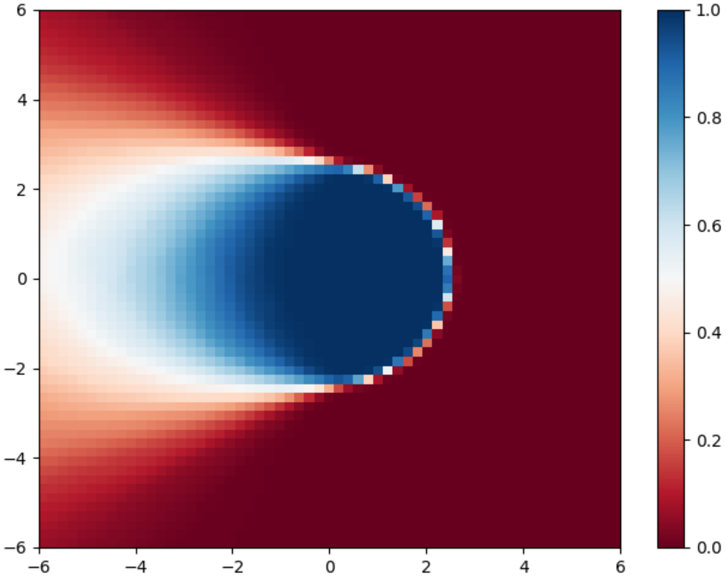
- M = 1.

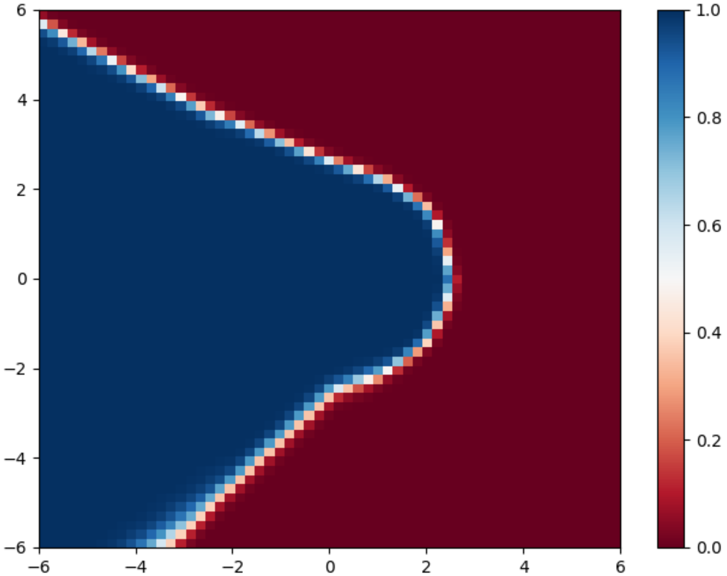




Toy classification problem – approach 4

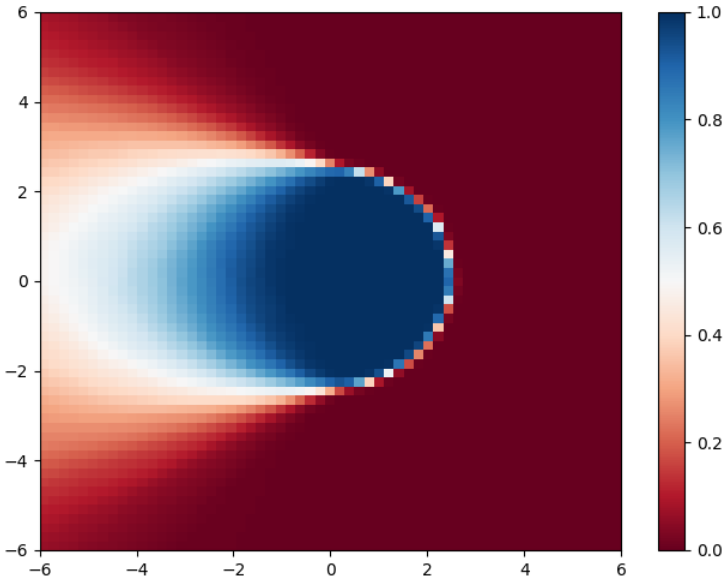
- M = 1.

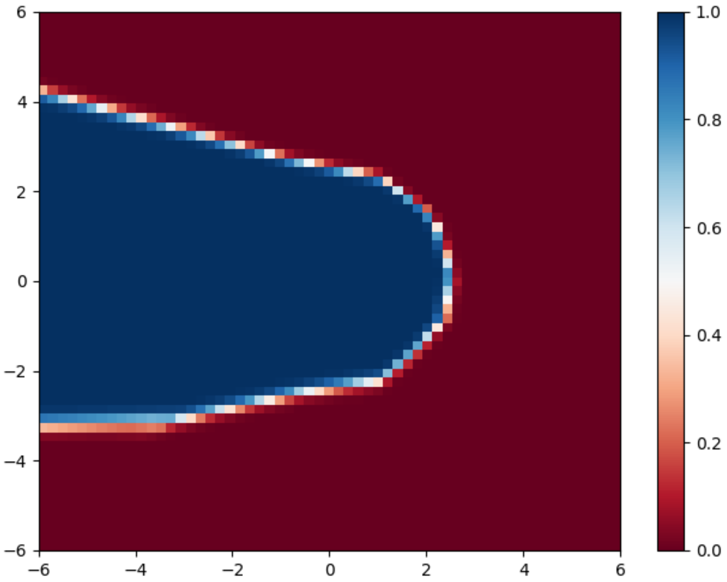




Toy classification problem – approach 4

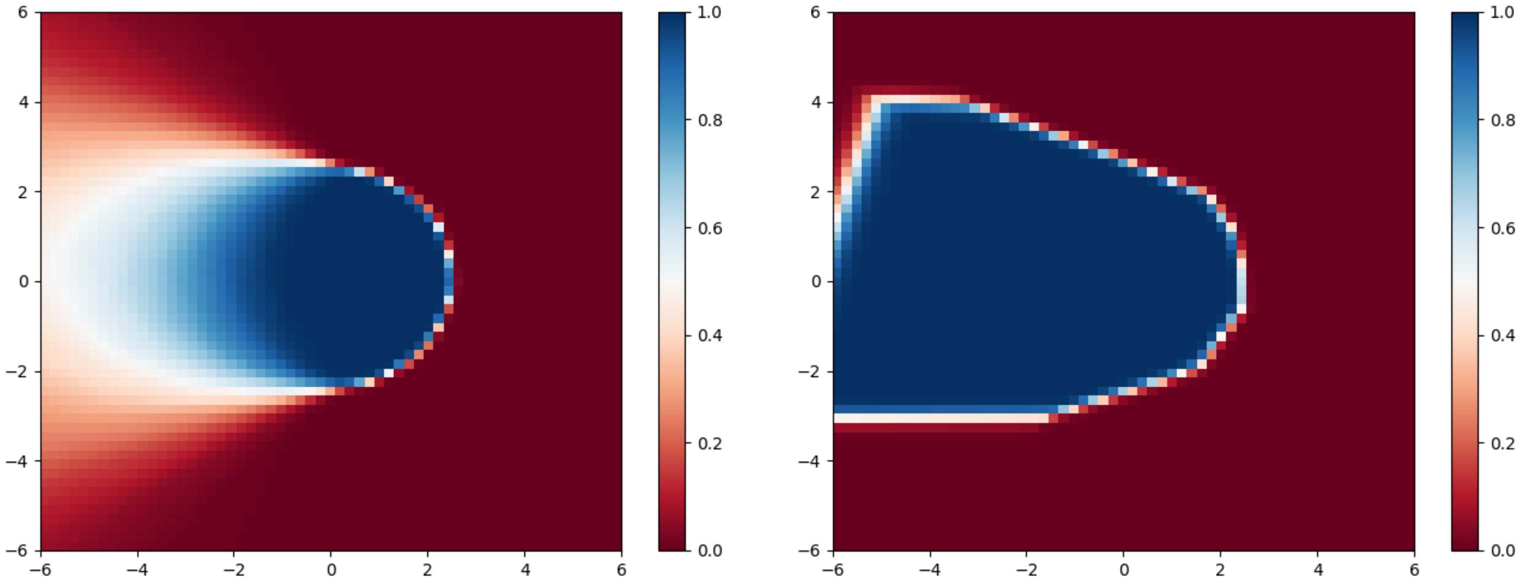
- M = 1.





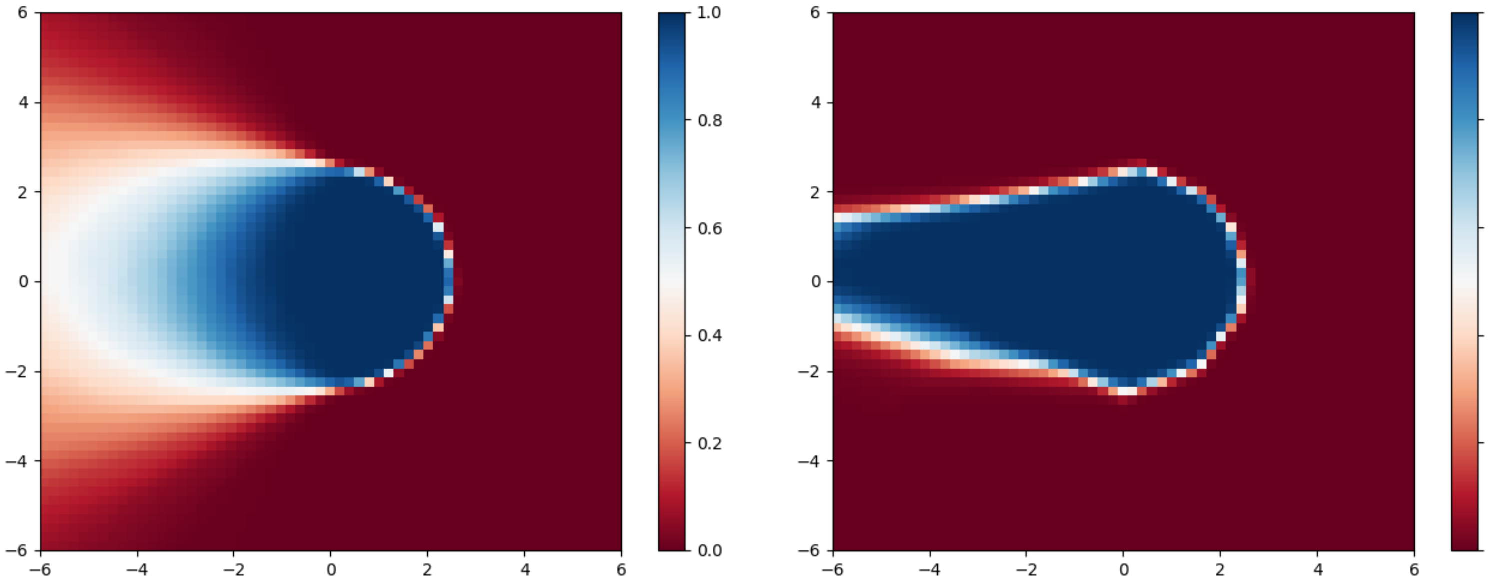
Toy classification problem – approach 4

- M = 1.



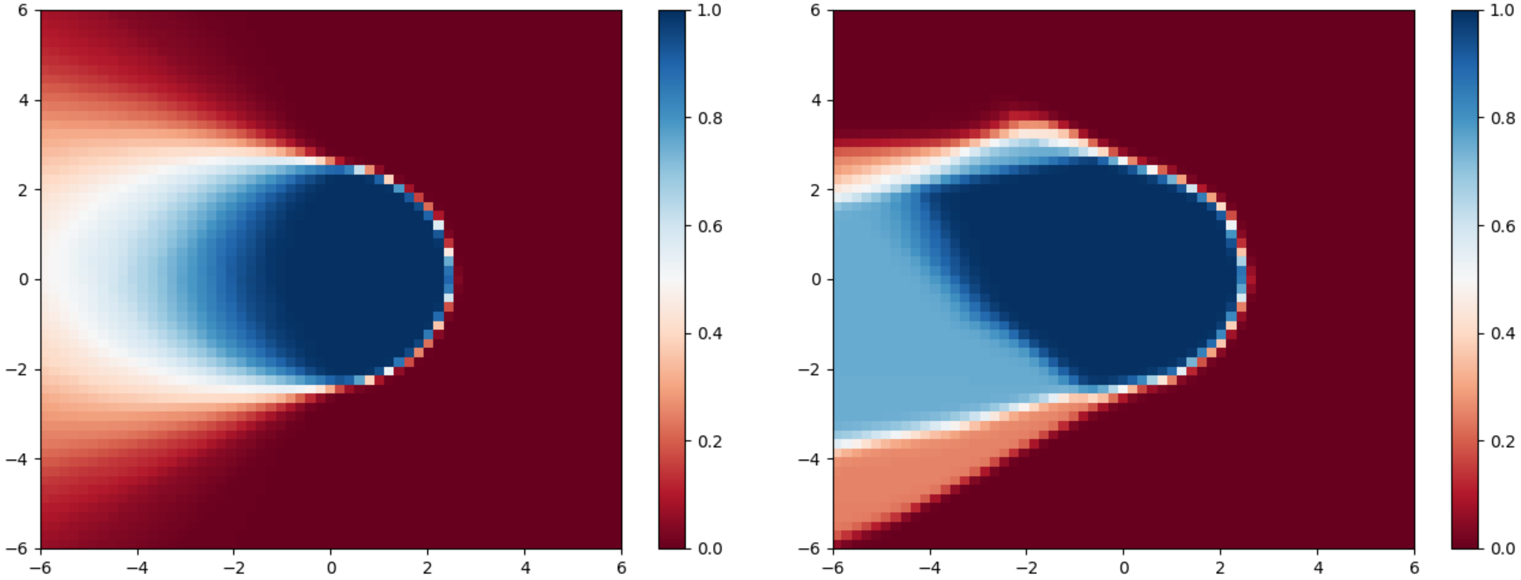
Toy classification problem – approach 4

- M = 1.



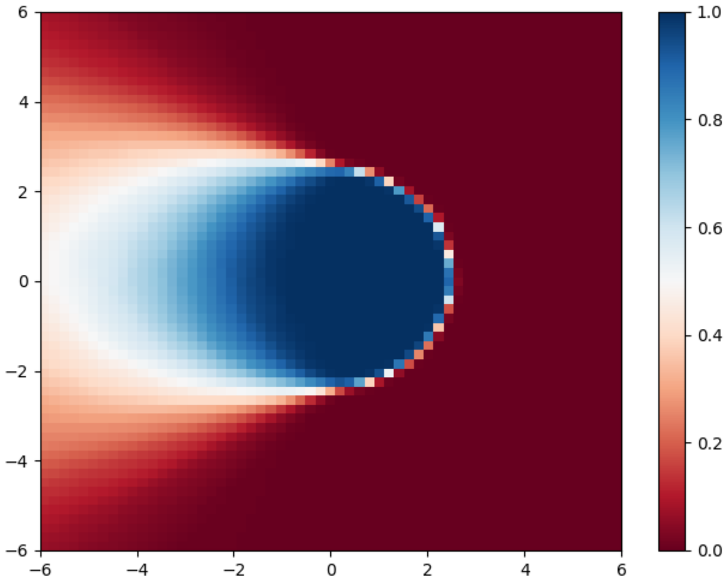
Toy classification problem – approach 4

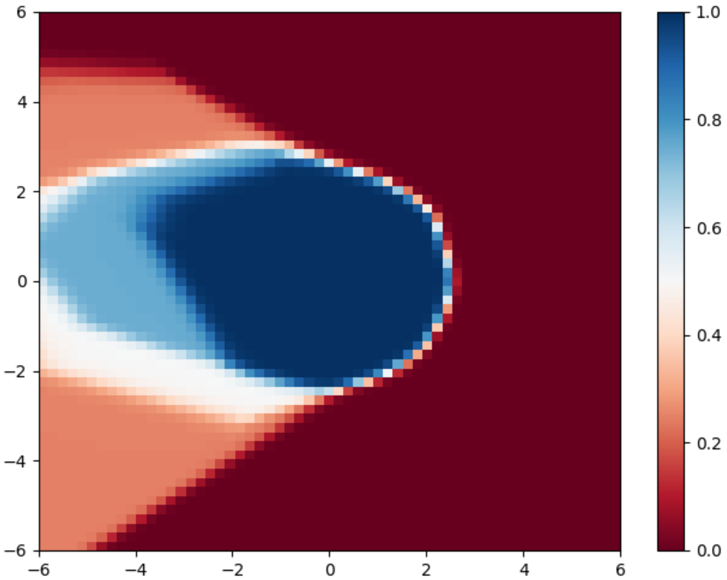
- M = 4.



Toy classification problem – approach 4

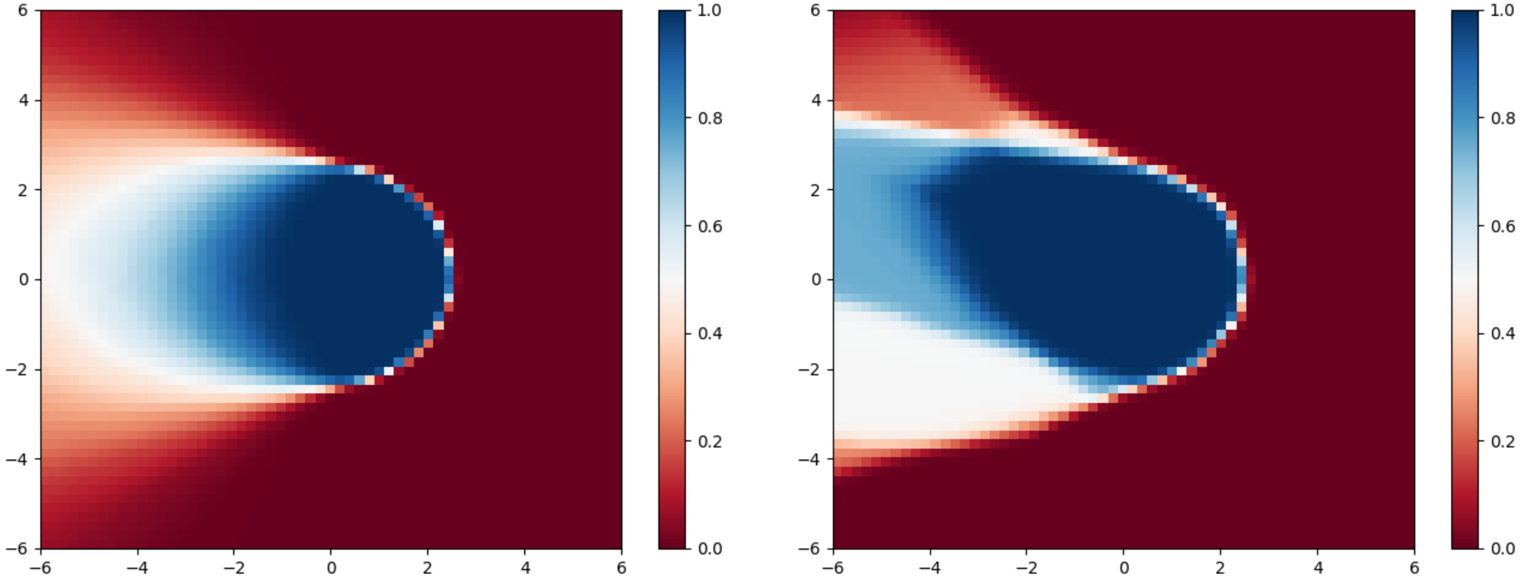
- M = 4.





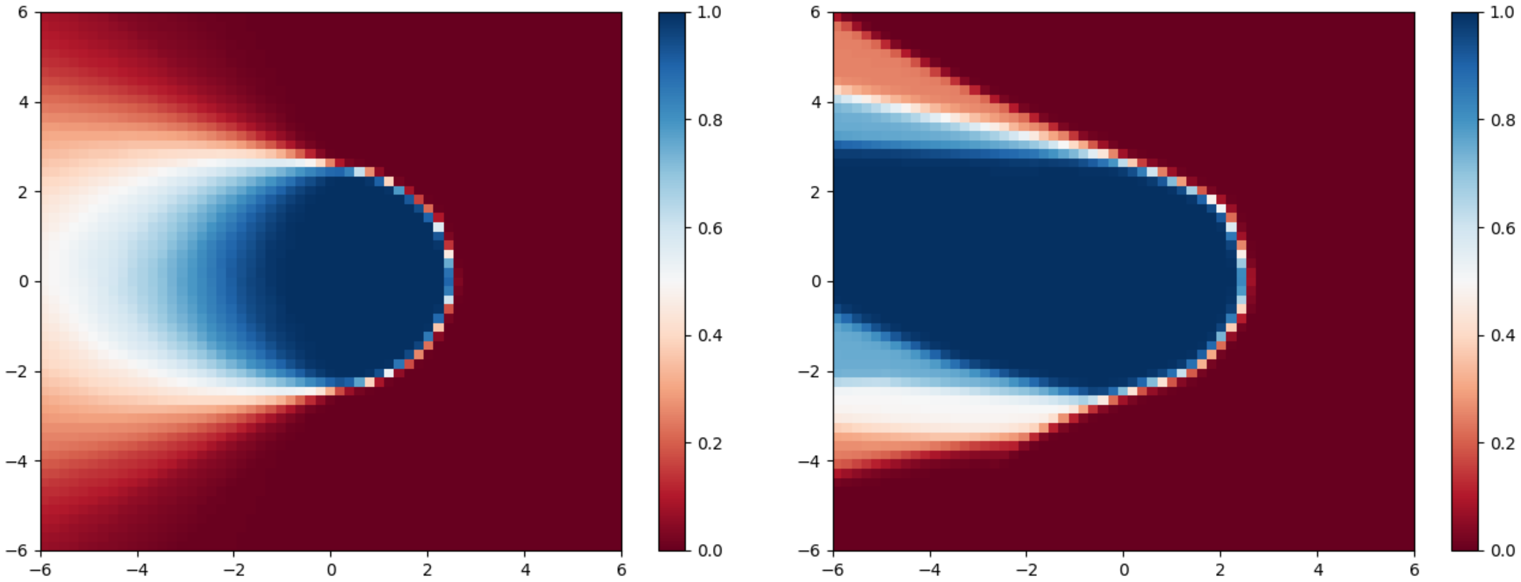
Toy classification problem – approach 4

- M = 4.



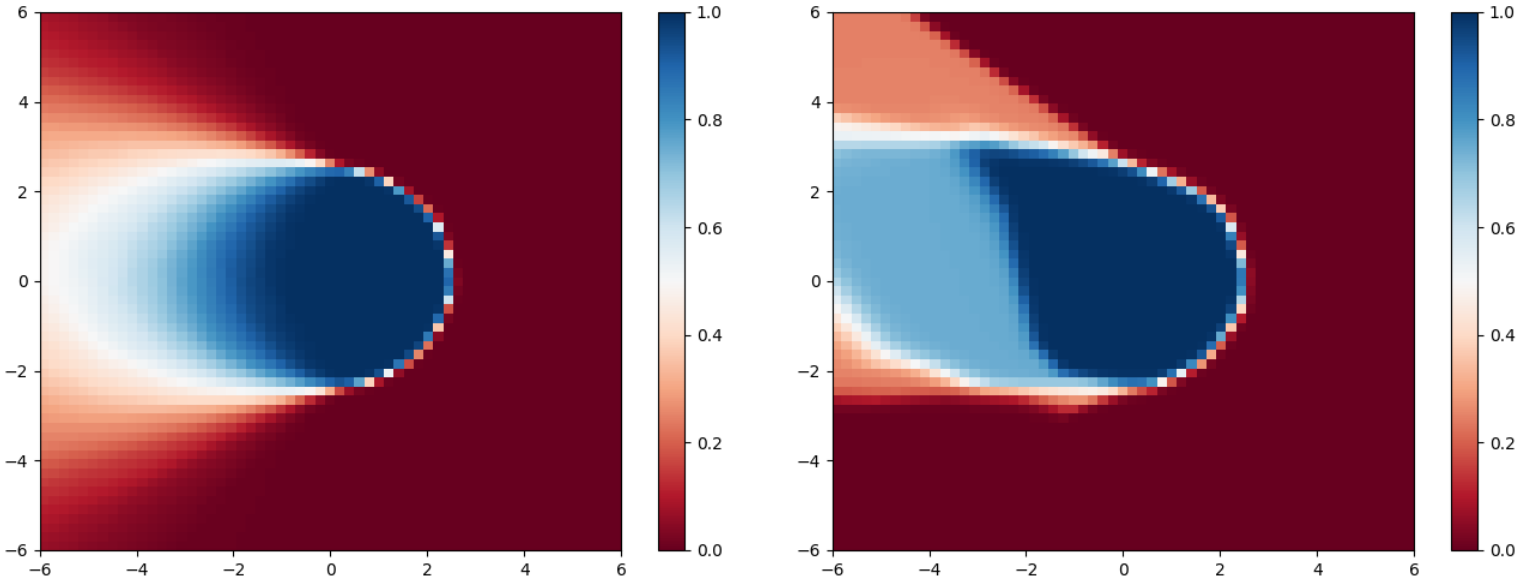
Toy classification problem – approach 4

- M = 4.



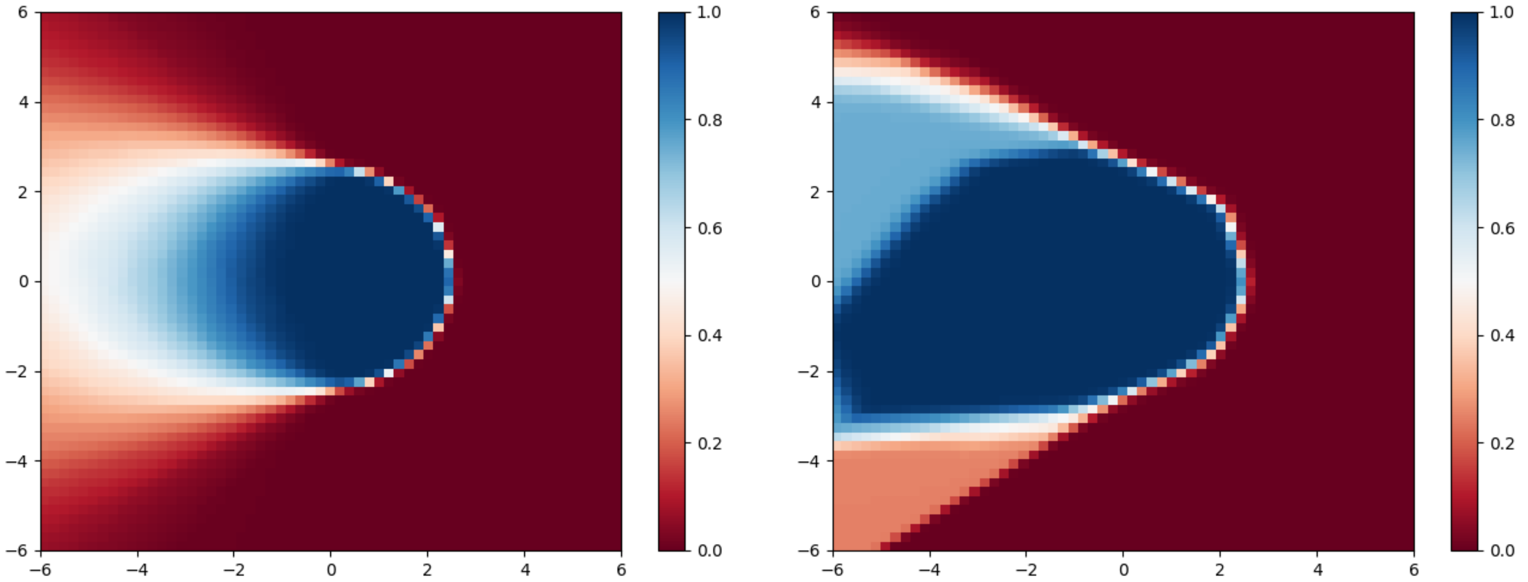
Toy classification problem – approach 4

- M = 4.



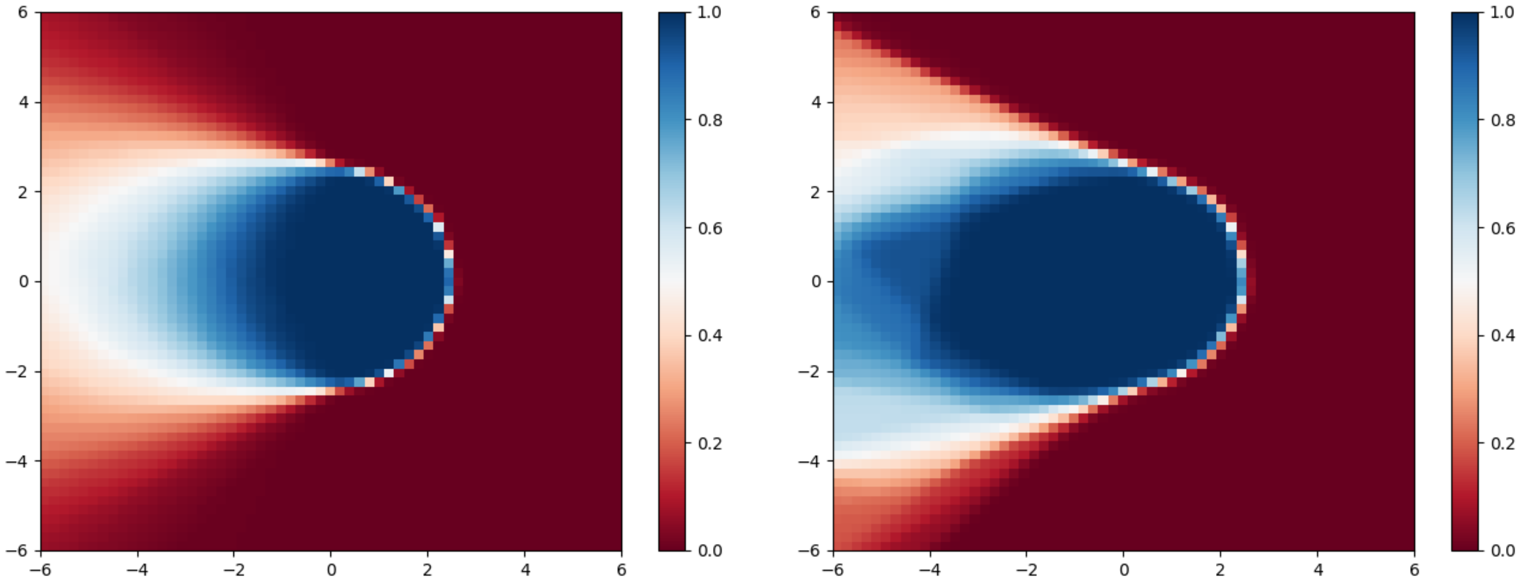
Toy classification problem – approach 4

- M = 4.



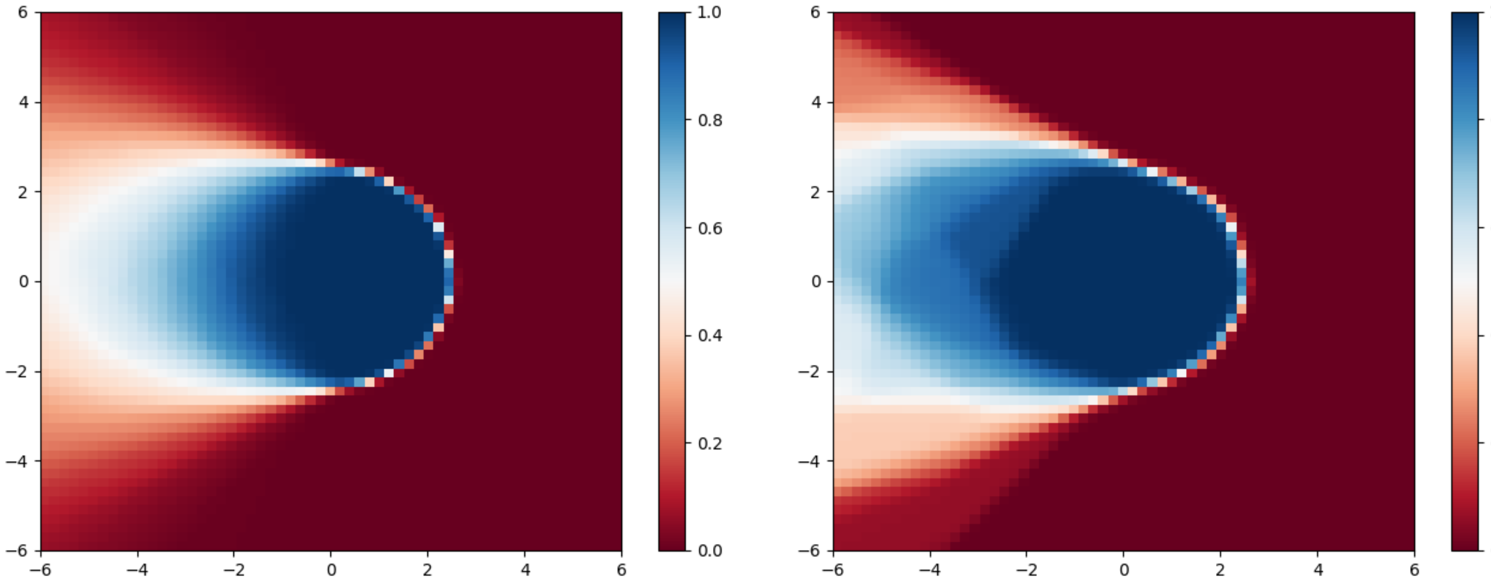
Toy classification problem – approach 4

- M = 16.



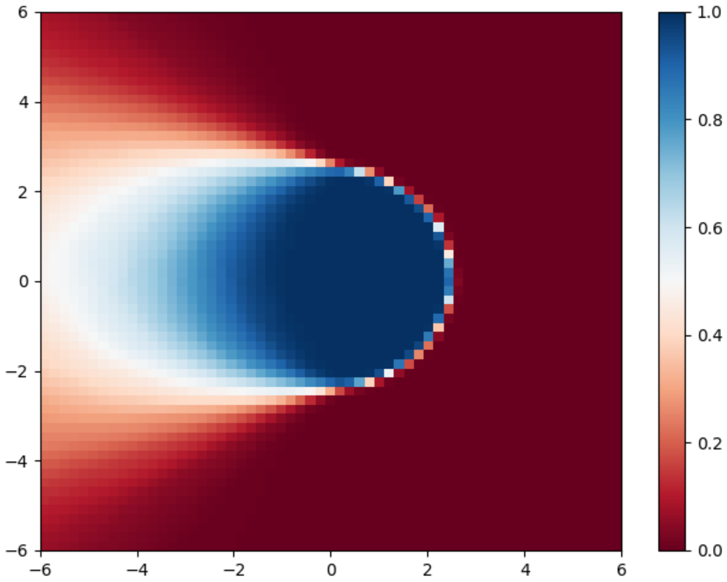
Toy classification problem – approach 4

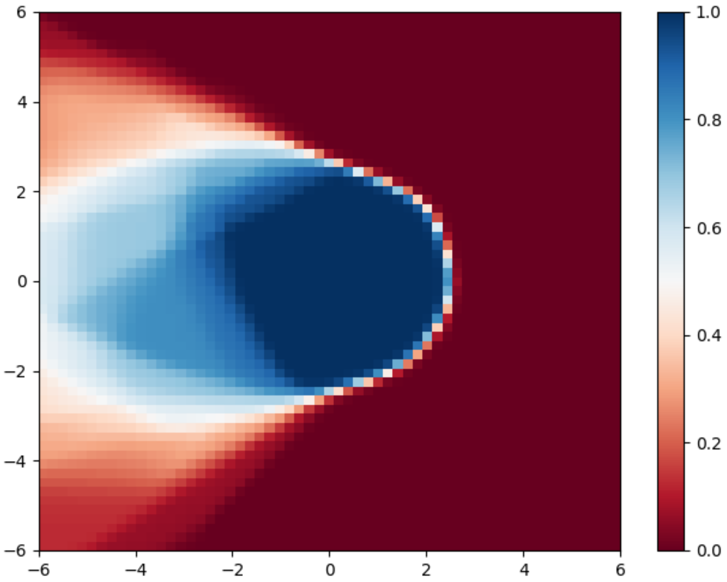
- M = 16.



Toy classification problem – approach 4

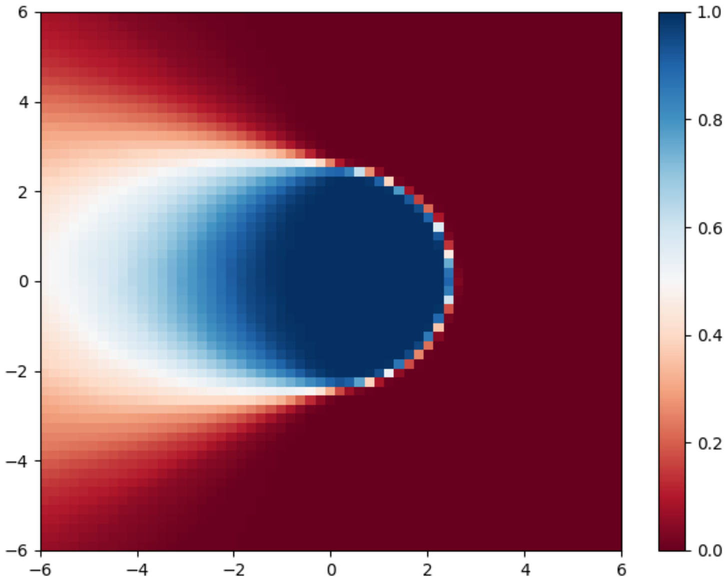
- M = 16.

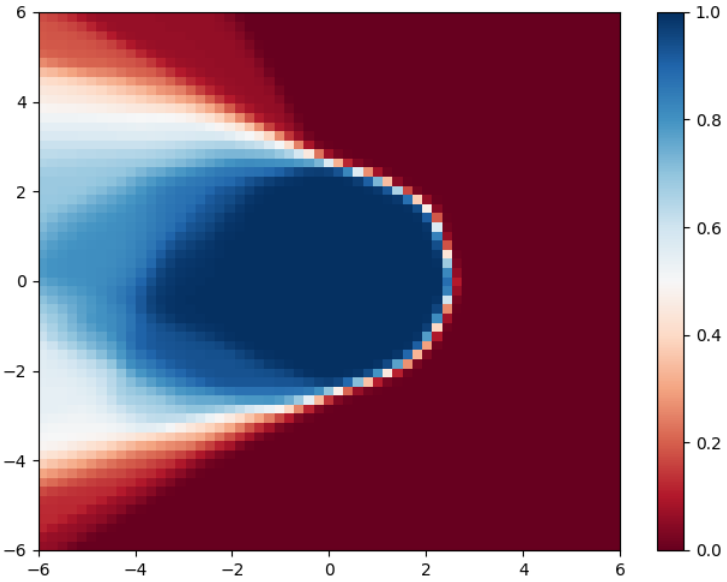




Toy classification problem – approach 4

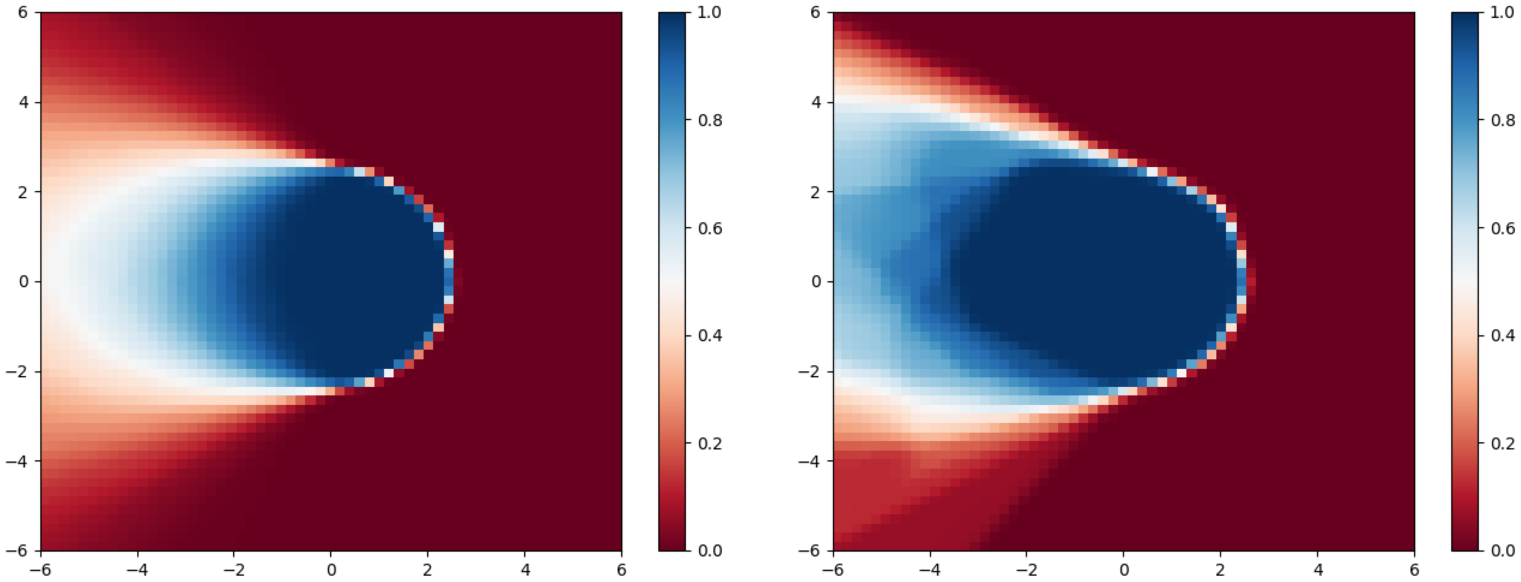
- M = 16.





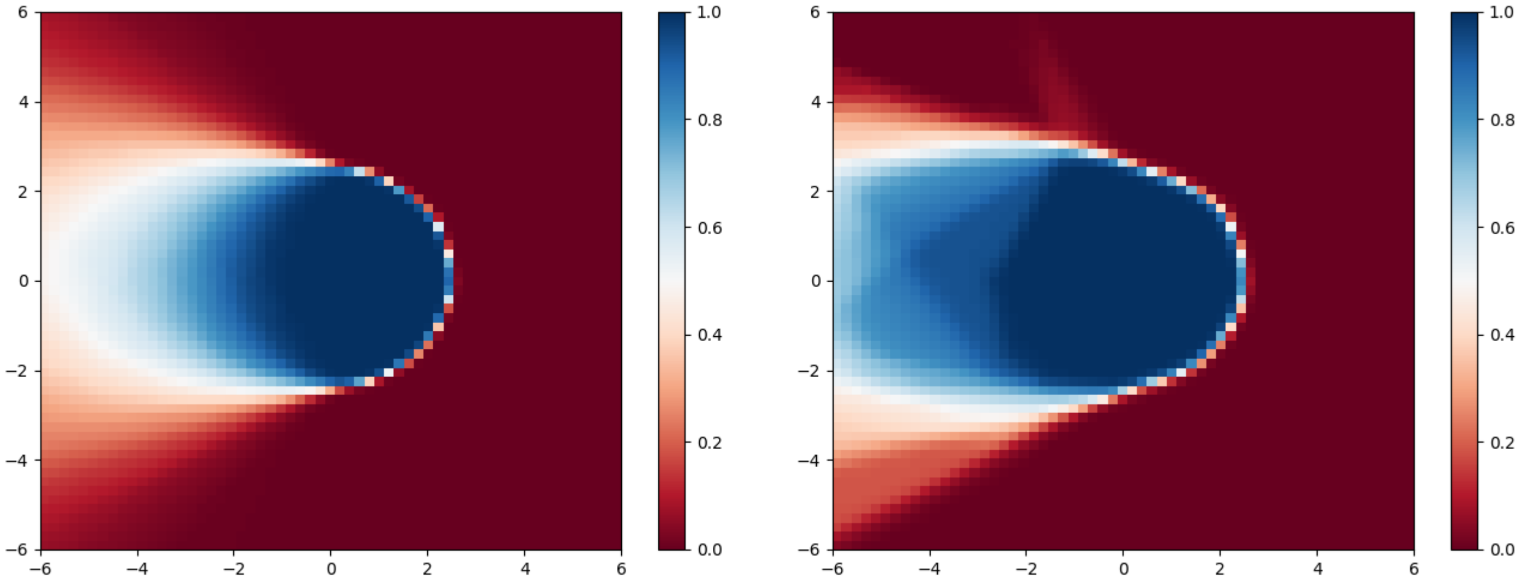
Toy classification problem – approach 4

- M = 16.



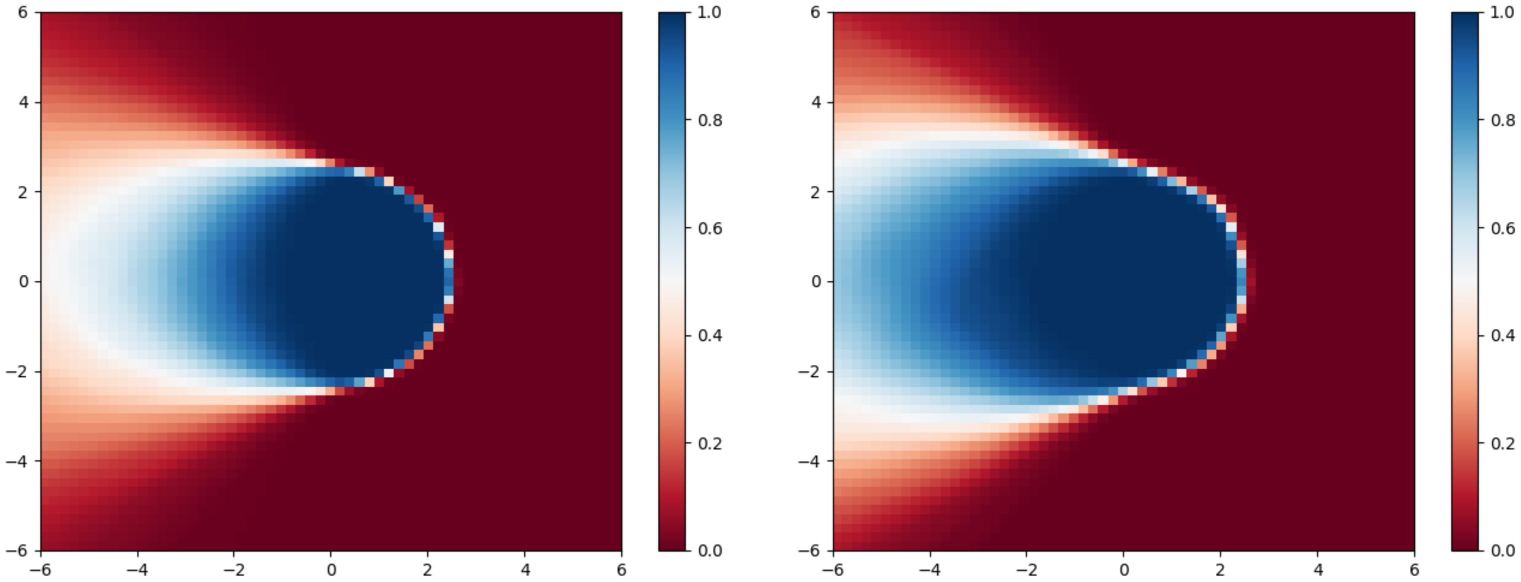
Toy classification problem – approach 4

- M = 16.



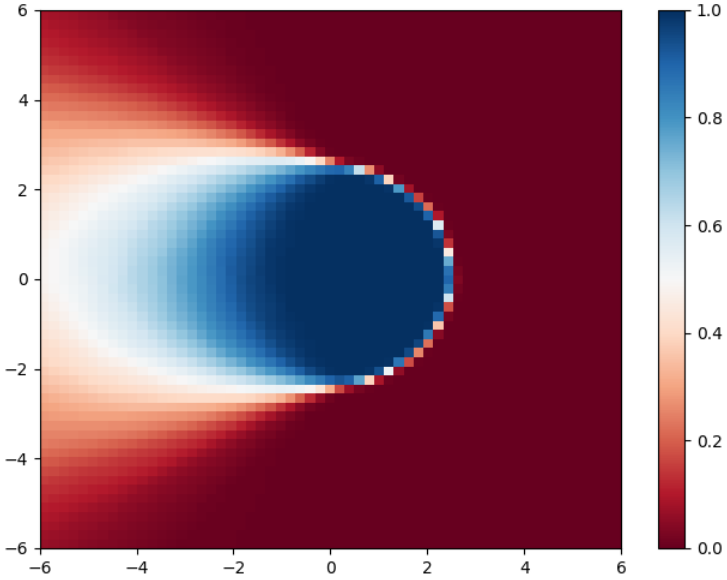
Toy classification problem – approach 4

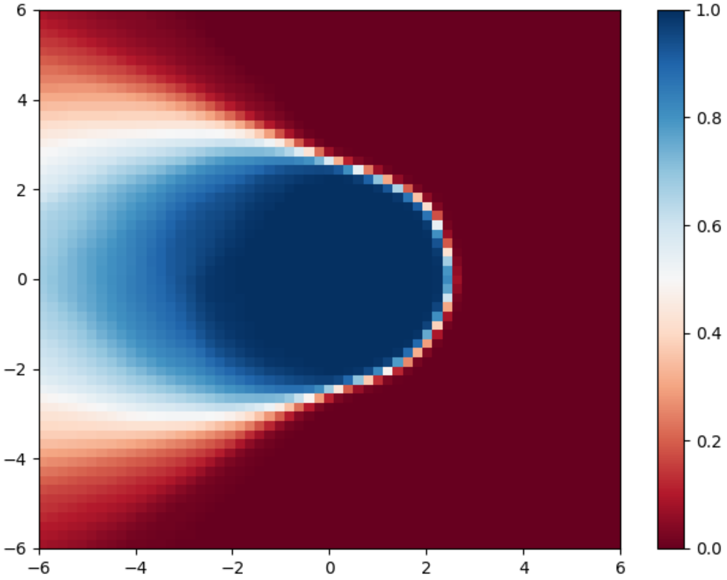
- M = 256.



Toy classification problem – approach 4

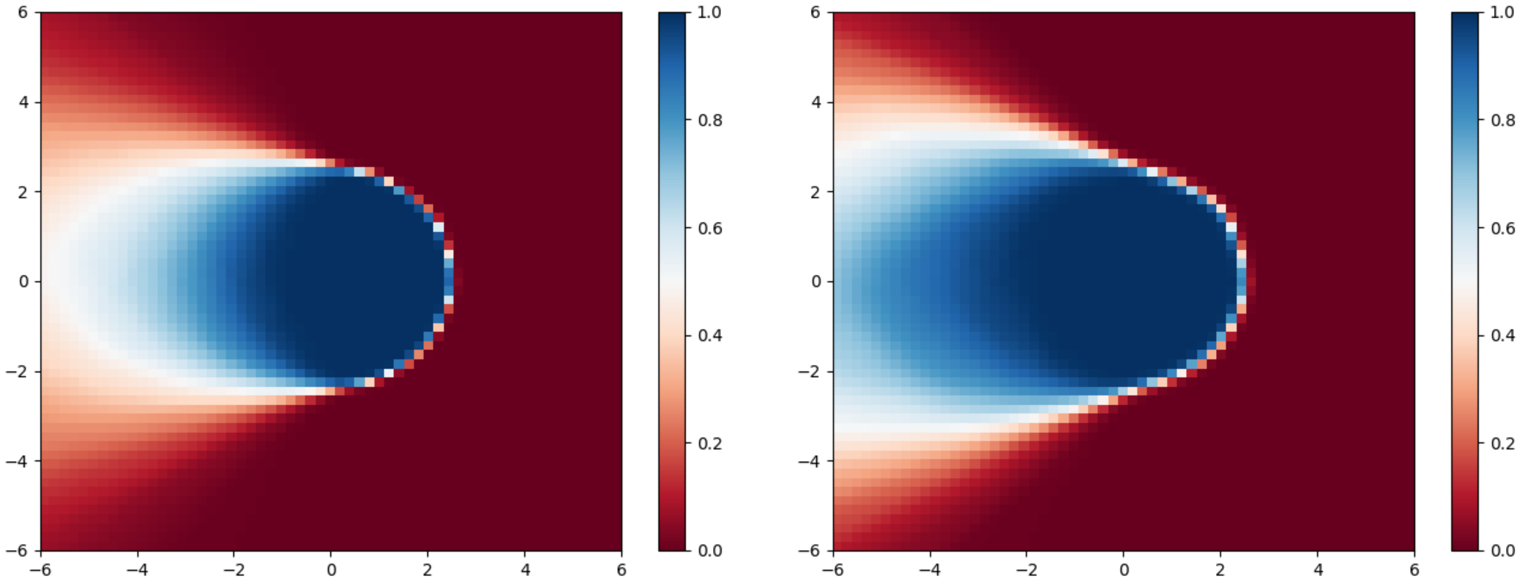
- M = 256.





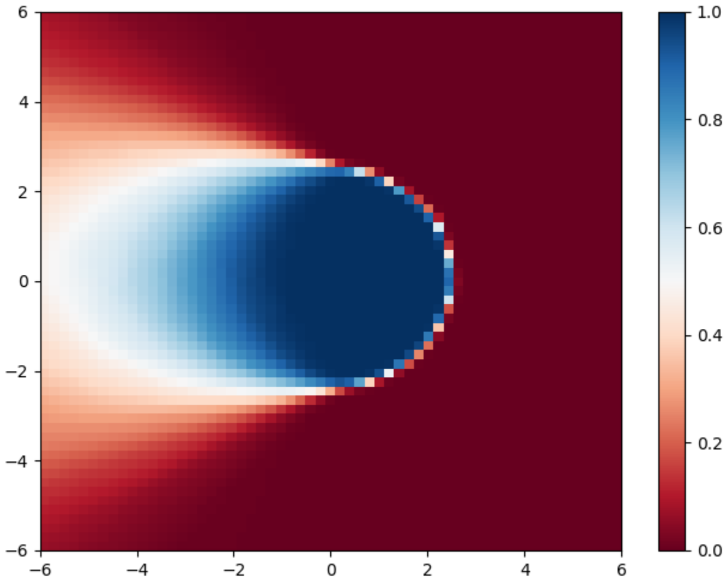
Toy classification problem – approach 4

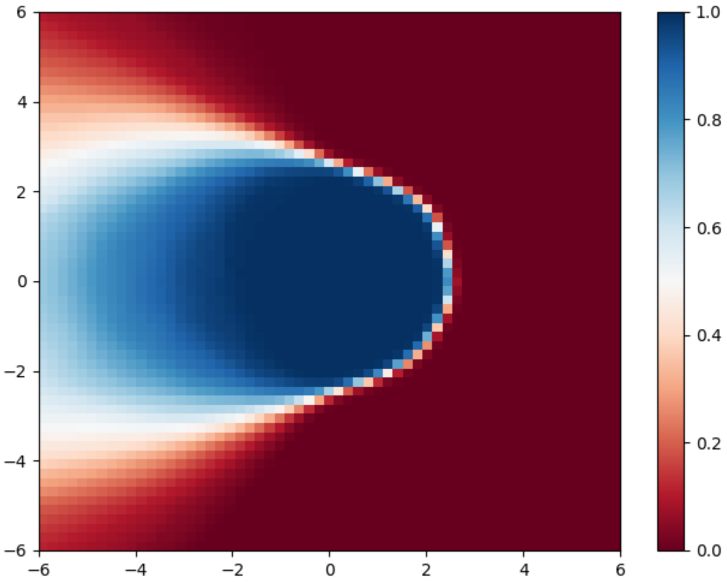
- M = 256.



Toy classification problem – approach 4

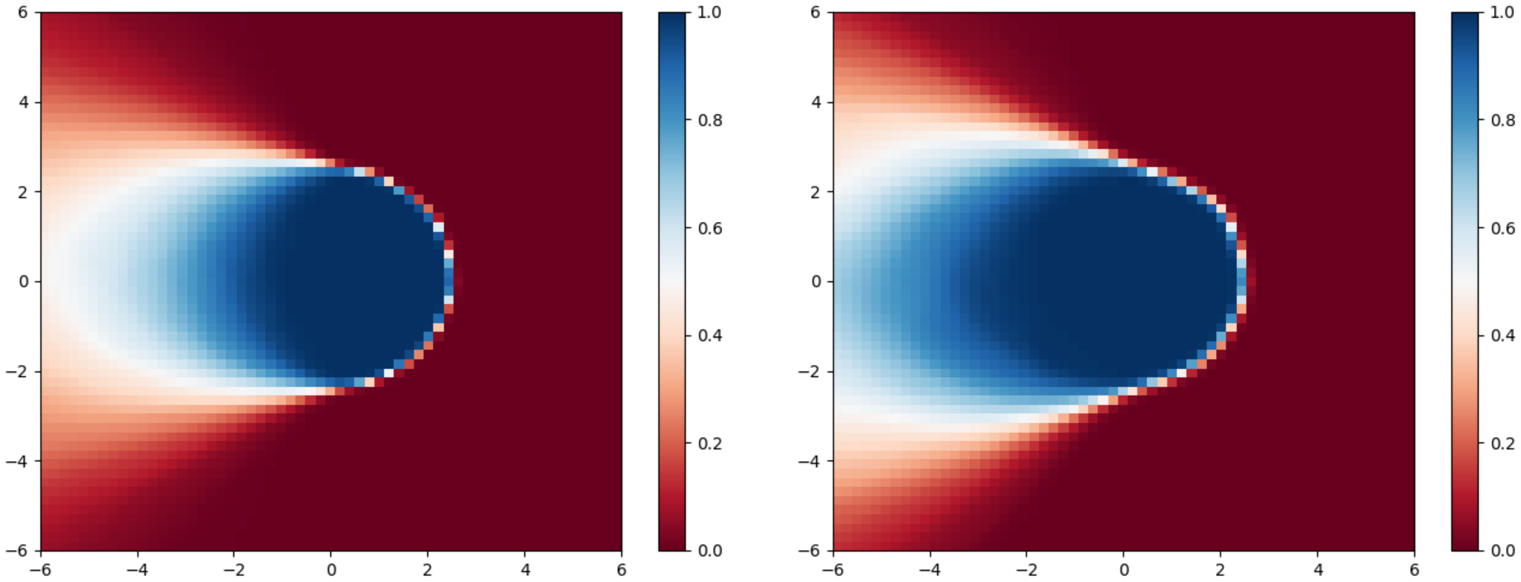
- M = 256.





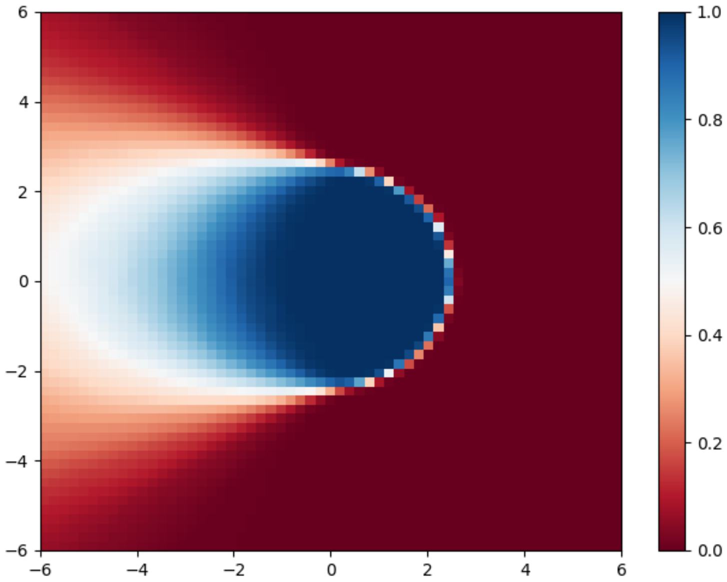
Toy classification problem – approach 4

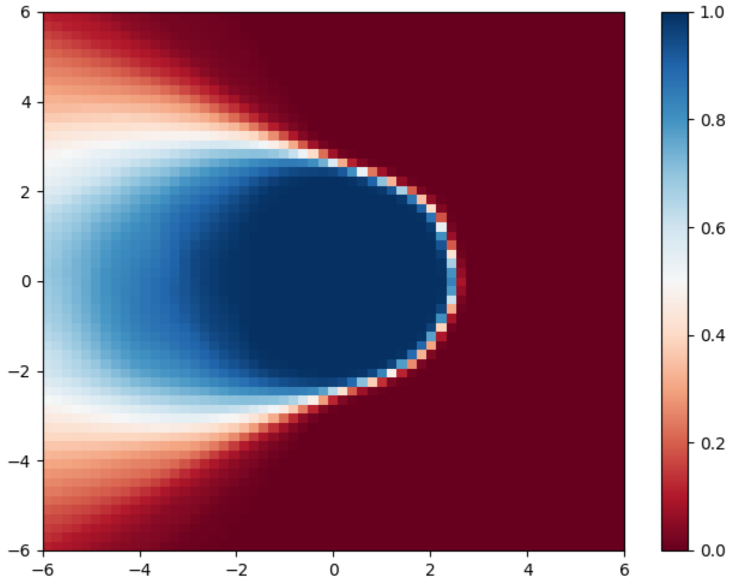
- M = 256.



Toy classification problem – approach 4

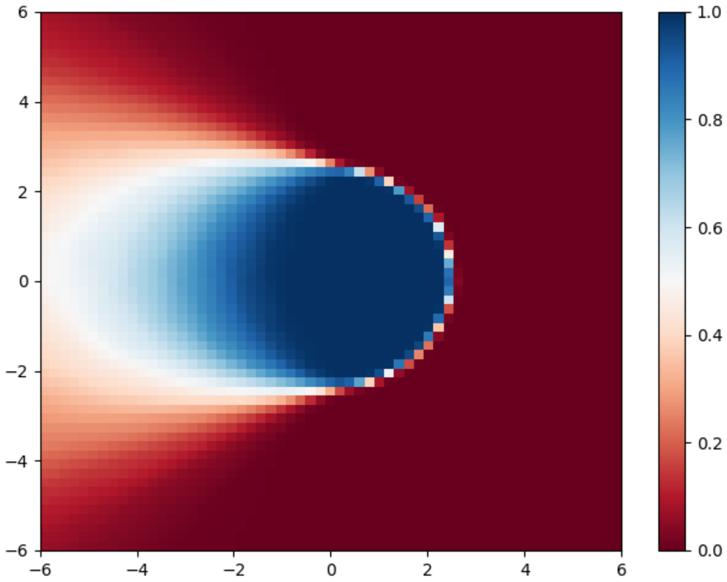
- M = 256.

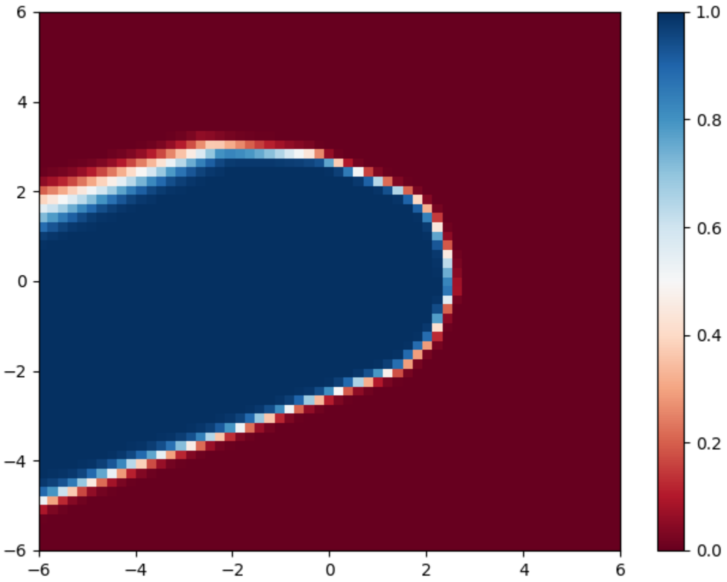




Toy classification problem – approach 4

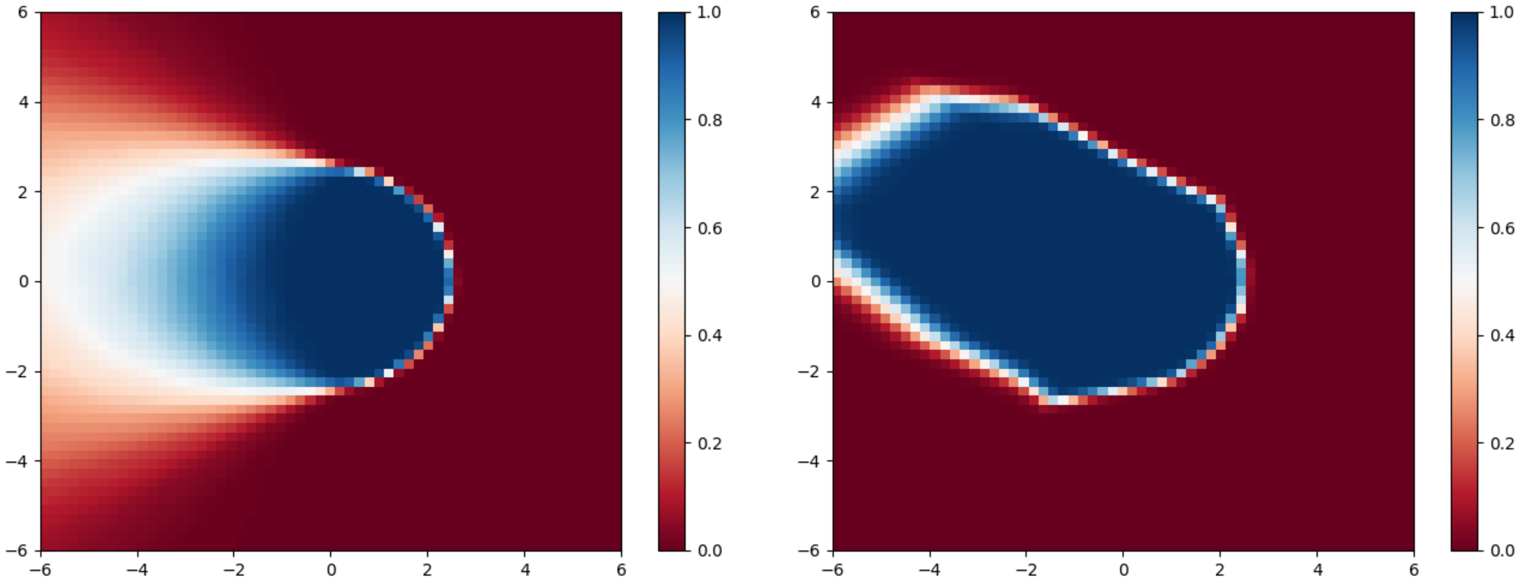
- M = 64, but all networks are trained with the **same** (randomly chosen) initialization.





Toy classification problem – approach 4

- M = 64, but all networks are trained with the **same** (randomly chosen) initialization.



Toy classification problem – approach 4

- M = 64, but all networks are trained with the **same** (randomly chosen) initialization.