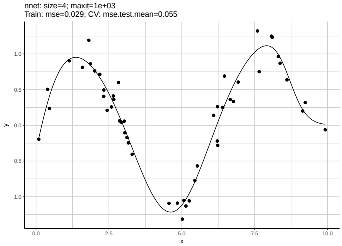
Deep Learning

Universal Approximation



Learning goals

- Universal approximation theorem for one-hidden-layer neural networks
- The pros and cons of a low approximation error

UNIVERSAL APPROXIMATION PROPERTY

Theorem. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a continuous, non-constant, bounded, and monotonically increasing function. Let $C \subset \mathbb{R}^p$ be compact, and let $\mathcal{C}(C)$ denote the space of continuous functions $C \to \mathbb{R}$. Then, given a function $g \in \mathcal{C}(C)$ and an accuracy $\varepsilon > 0$, there exists a hidden layer size $m \in \mathbb{N}$ and a set of coefficients $W_j \in \mathbb{R}^p$, $u_j, b_j \in \mathbb{R}$ (for $j \in \{1, \ldots, m\}$), such that

$$f : C \to \mathbb{R}\,; \quad f(\mathbf{x}) = \sum_{j=1}^{m} u_j \cdot \sigma\Big(W_j^T \mathbf{x} + b_j\Big)$$

is an ε-approximation of g, that is,

$$\|f - g\|_\infty := \max_{x \in C} |f(\mathbf{x}) - g(\mathbf{x})| < \varepsilon \ \ .$$

The theorem extends trivially to multiple outputs.

UNIVERSAL APPROXIMATION PROPERTY

Corollary. Neural networks with a single sigmoidal hidden layer and linear output layer are universal approximators.

- This means that for a given target function g there exists a sequence of networks $\left(f_k\right)_{k\in\mathbb{N}}$ that converges (pointwise) to g.
- Usually, as the networks come closer and closer to g, they will need more and more hidden neurons.
- A network with fixed layer sizes can only model a subspace of all continuous functions.
- The continuous functions form an infinite dimensional vector space. Therefore arbitrarily large hidden layer sizes are needed.

UNIVERSAL APPROXIMATION PROPERTY

- Why is universal approximation a desirable property?
- Recall the definition of a Bayes optimal hypothesis $h^* : X \to Y$. It is the best possible hypothesis (model) for the given problem: it has minimal loss averaged over the data generating distribution.
- So ideally we would like the neural network (or any other learner) to approximate the Bayes optimal hypothesis.
- Usually we do not manage to learn h^*.
- This is because we do not have enough (infinite) data. We have no control over this, so we have to live with this limitation.
- But we do have control over which model class we use.

UNIVERSAL APPROXIMATION PROPERTY

- Universal approximation ⇒ approximation error tends to zero as hidden layer size tends to infinity.

- Positive approximation error implies that no matter how big the data set, we cannot find the optimal model.

- This bears the risk of systematic under-fitting, which can be avoided with a universal model class.

UNIVERSAL APPROXIMATION PROPERTY

- As we know, there are also good reasons for restricting the model class.

- This is because a flexible model class with universal approximation ability often results in over-fitting, which is no better than under-fitting.

- Thus, “universal approximation $\Rightarrow$ low approximation error”, but at the risk of a substantial learning error.

- In general, models of intermediate flexibility give the best predictions. For neural networks this amounts to a reasonably sized hidden layer.

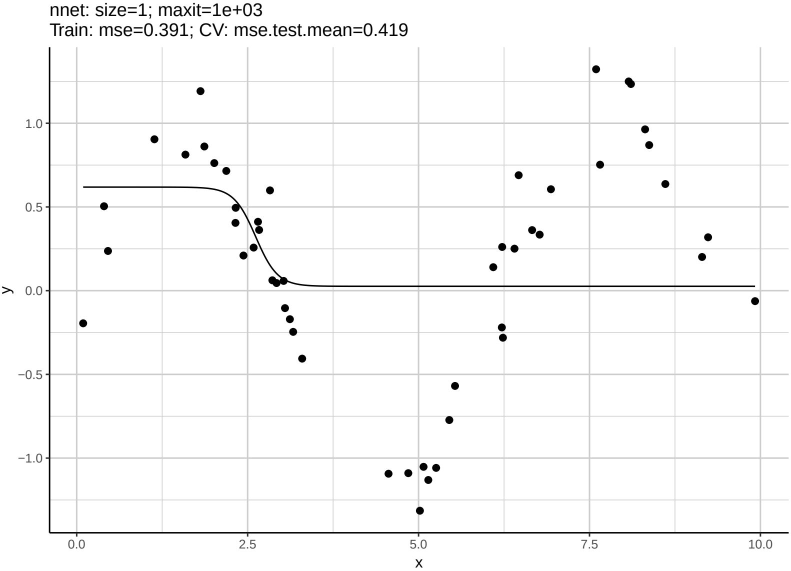
EXAMPLE : REGRESSION/CLASSIFICATION

- Let’s look at a few examples of the types of functions and decisions boundaries learnt by neural networks (with a **single** hidden layer) of various sizes.

- "size" here refers to the number of neurons in the hidden layer.

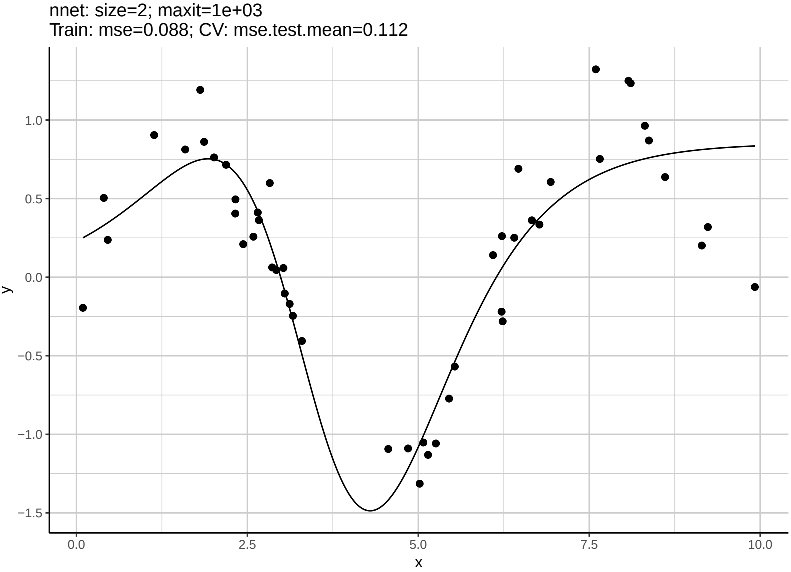
- The number of "iterations" in the following slides corresponds to the number of steps of the applied iterative optimization algorithm (stochastic gradient descent).

REGRESSION EX.: 1000 TRAINING ITERATIONS

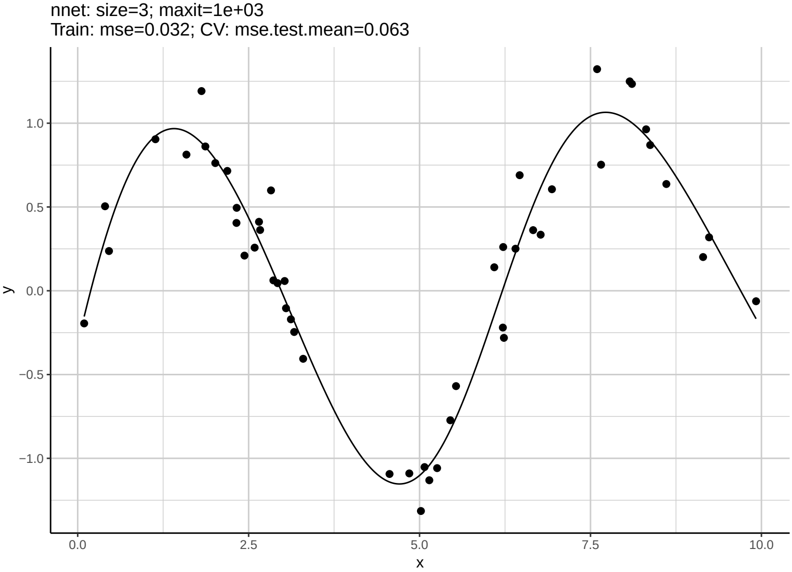


nnet: size=1; maxit=1e+03
Train: mse=0.391; CV: mse.test.mean=0.419

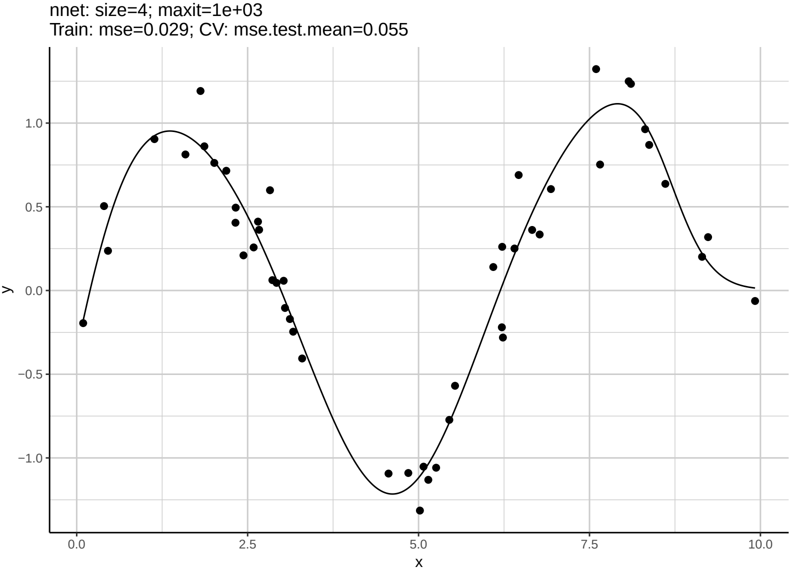
REGRESSION EX.: 1000 TRAINING ITERATIONS



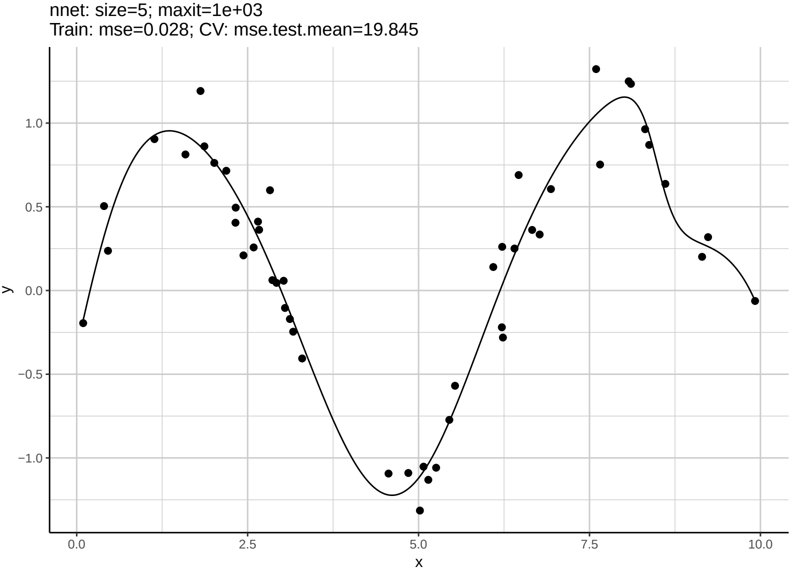
REGRESSION EX.: 1000 TRAINING ITERATIONS



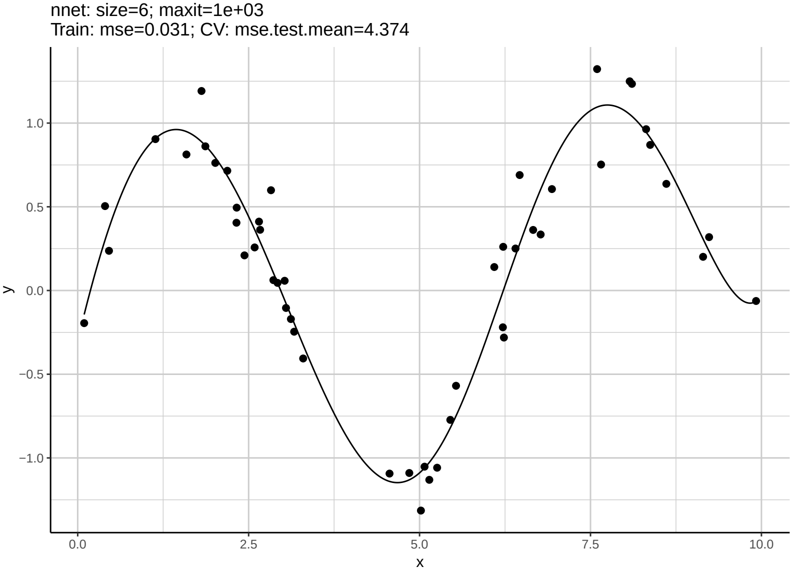
REGRESSION EX.: 1000 TRAINING ITERATIONS



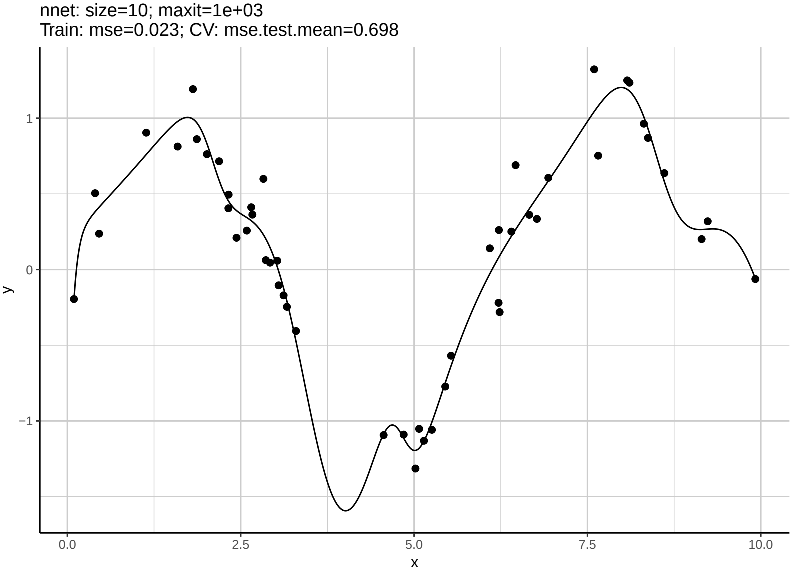
REGRESSION EX.: 1000 TRAINING ITERATIONS



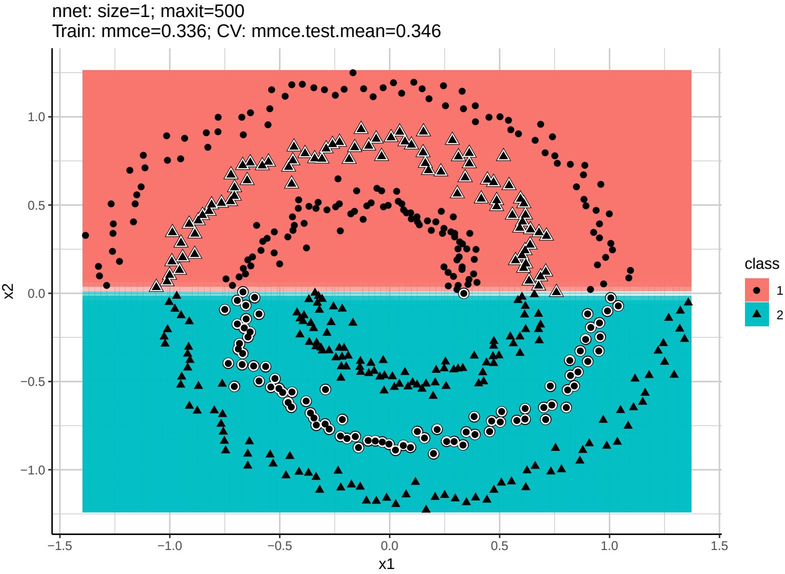
REGRESSION EX.: 1000 TRAINING ITERATIONS



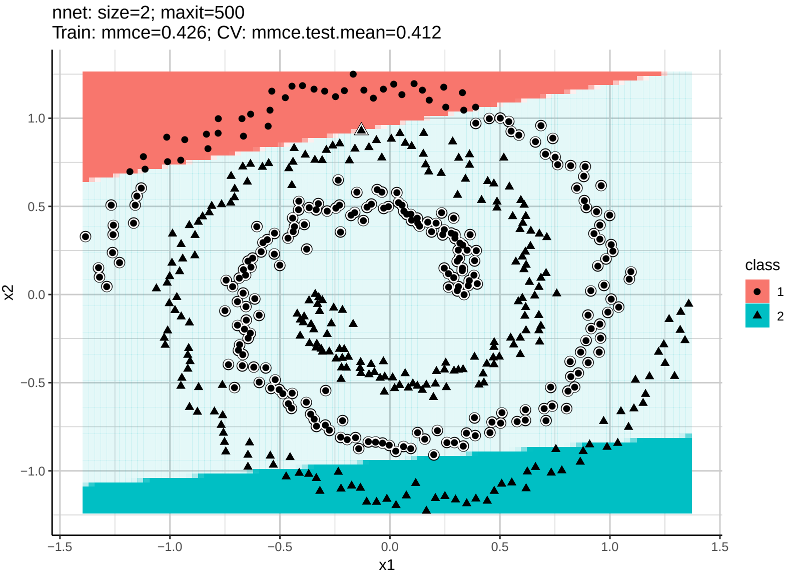
REGRESSION EX.: 1000 TRAINING ITERATIONS



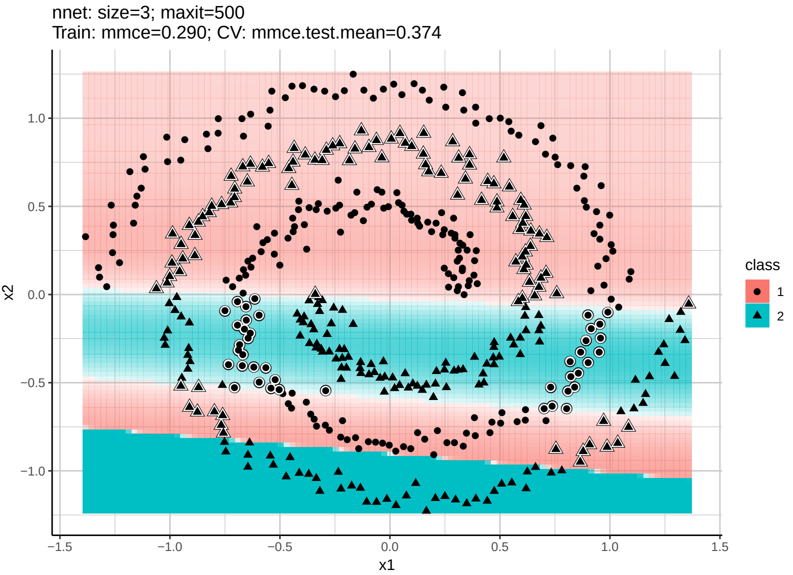
CLASSIFICATION: 500 TRAINING ITERATIONS



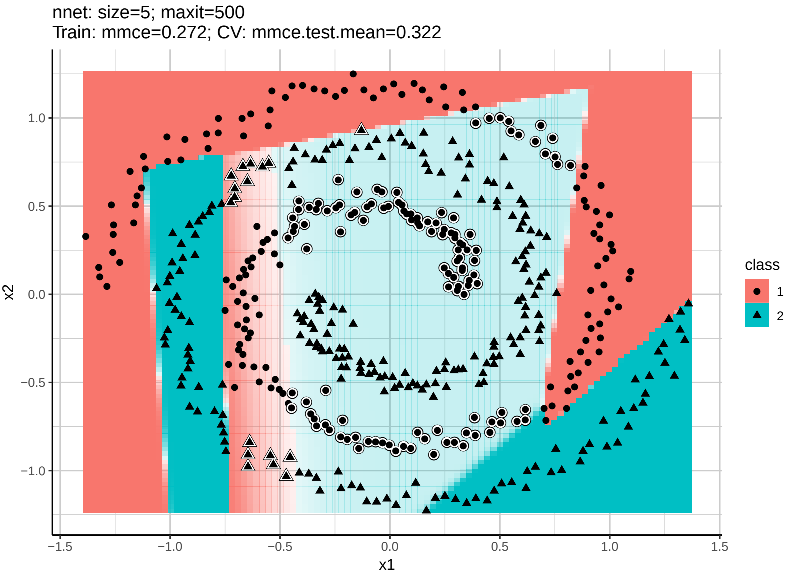
CLASSIFICATION: 500 TRAINING ITERATIONS



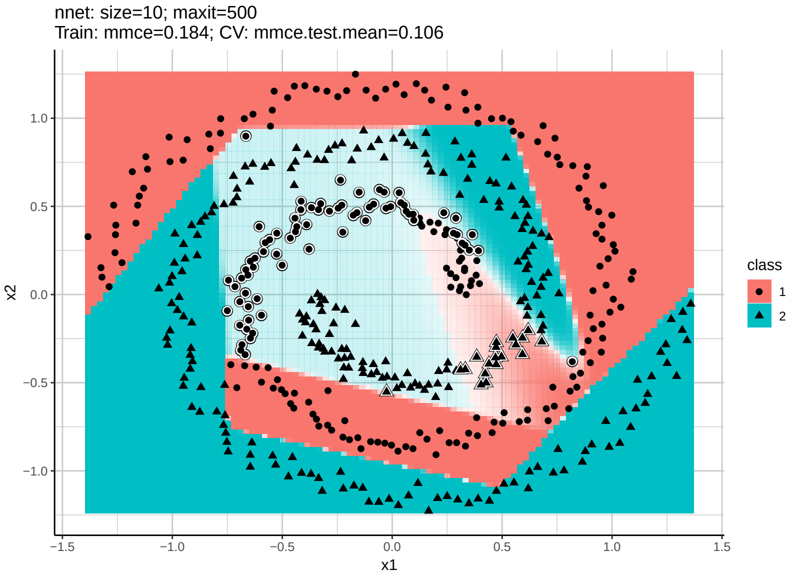
CLASSIFICATION: 500 TRAINING ITERATIONS



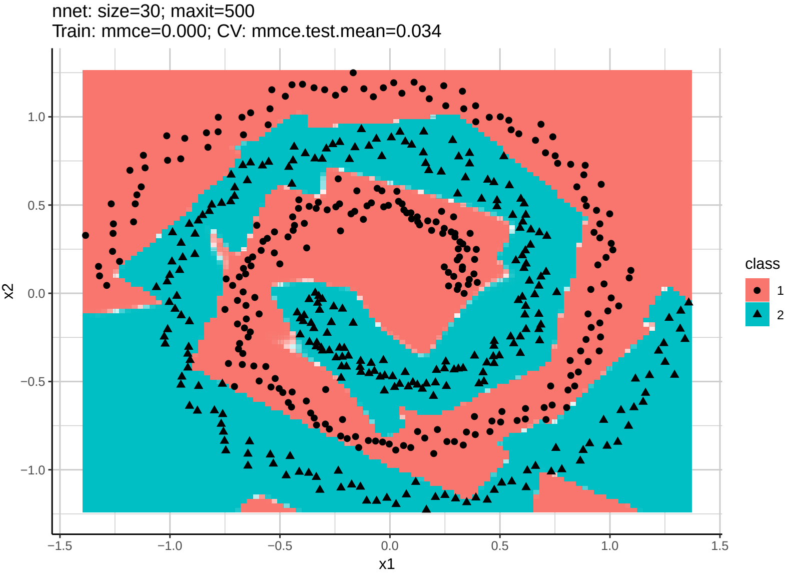
CLASSIFICATION: 500 TRAINING ITERATIONS



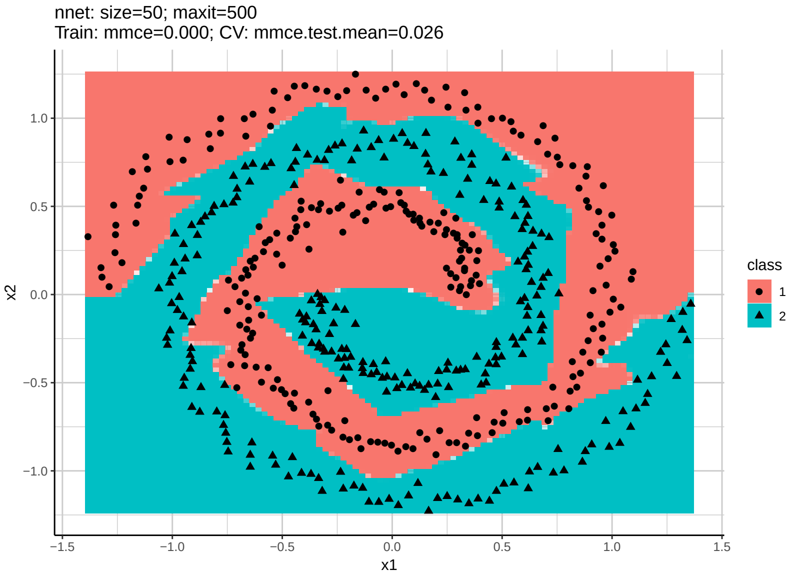
CLASSIFICATION: 500 TRAINING ITERATIONS



CLASSIFICATION: 500 TRAINING ITERATIONS



CLASSIFICATION: 500 TRAINING ITERATIONS


nnet: size=50; maxit=500
Train: mmce=0.000; CV: mmce.test.mean=0.026
x2
x1
class
● 1
▲ 2