Machine Intelligence 1 — **Exercise Sheet 7**
WS 2022/23, Obermayer/Kashef/Strömsdörfer — due: 05.01.2023 at 23:55

Radial basis function networks

Exercise T7.1: Multi-class classification: simple methods (tutorial)

(a) Describe how a k nearest neighbor classifier predicts the class of previously unseen inputs?

(b) A "Parzen window" classifier extends the *electoral committee* approach of kNN. How are the different votes *weighted*?

Exercise T7.2: Radial basis function networks (tutorial)

(a) Describe and discuss the *general architecture* of an RBF-network.

(b) Describe and discuss the *two-step learning procedure* for RBF-networks with K basis functions.

(c) In which cases of *regression* or *classification* do RBF networks outperform *MLPs* significantly? What are the advantages of RBF networks? In which situations are they preferable to MLP?

Exercise H7.1: Training data (homework, 1 point)

Create a sample of $p = 120$ training patterns $\{\underline{\mathbf{x}}^{(\alpha)}, y_T^{(\alpha)}\}$, $\alpha = 1, \ldots, p$. The input values $\underline{\mathbf{x}}^{(\alpha)} \in \mathbb{R}^2$ should be drawn from a mixture of Gaussians with centers in an XOR-configuration according to the following scheme:

- Generate 60 samples from each of the following conditional distributions:

$$p(\underline{\mathbf{x}}|y=0) \quad := \tfrac{1}{2}\left[\mathcal{N}(\underline{\mathbf{x}}|\underline{\boldsymbol{\mu}}_1, \underline{\mathbf{I}}\,\sigma^2) + \mathcal{N}(\underline{\mathbf{x}}|\underline{\boldsymbol{\mu}}_2, \underline{\mathbf{I}}\,\sigma^2)\right],$$
$$p(\underline{\mathbf{x}}|y=1) \quad := \tfrac{1}{2}\left[\mathcal{N}(\underline{\mathbf{x}}|\underline{\boldsymbol{\mu}}_3, \underline{\mathbf{I}}\,\sigma^2) + \mathcal{N}(\underline{\mathbf{x}}|\underline{\boldsymbol{\mu}}_4, \underline{\mathbf{I}}\,\sigma^2)\right],$$

  with $\underline{\boldsymbol{\mu}}_1 = (0,1)^\top, \underline{\boldsymbol{\mu}}_2 = (1,0)^\top, \underline{\boldsymbol{\mu}}_3 = (0,0)^\top, \underline{\boldsymbol{\mu}}_4 = (1,1)^\top$
  and a variance of $\sigma^2 = 0.1$.

  To sample from one of the two mixture variables you can

  (i) draw with probability 1/2 whether you will draw the next point from the density in the left summand or from the density in the right summand, then

  (ii) sample from that normal distribution yielding a single point $\underline{\mathbf{x}}^{(\alpha)}$.

  Note that $\mathcal{N}(\underline{\mathbf{x}}|\underline{\boldsymbol{\mu}}, \underline{\mathbf{I}}\,\sigma^2)$ is the probability density of a multivariate normal distribution of a vector $\underline{\mathbf{x}}$, where $\underline{\boldsymbol{\mu}}$ is the mean vector and σ^2 the variance. The variance is the same for all components.

- The corresponding target values $y_T^{(\alpha)} \in \{0, 1\}$ describe the assignment to the two classes and indicate from which distribution [$p(\underline{\mathbf{x}}|y=0)$ vs. $p(\underline{\mathbf{x}}|y=1)$] the data point was drawn.

(a) (1 point) Plot the resulting 120 input samples $\underline{\mathbf{x}}^{(\alpha)}$ in a scatter plot, in which the markers and/or colors represent the corresponding samples' labels $y_T^{(\alpha)}$.

Exercise H7.2: k nearest neighbors (kNN) (homework, 2 points)

Build a kNN classifier that classifies new data (*query points*) by voting of the k nearest neighbors from the training set. The *electoral committee* is selected from the training patterns according to their Euclidean distance to the query point. The predicted class is determined by the target value of the majority of those k nearest patterns.

(a) (2 points) Plot the training patterns and the decision boundary (e.g. using a contour plot or a high-resolution image of)[1] in input space for $k = 1, 3, 5$. What do you observe?

Exercise H7.3: "Parzen window" classifier (homework, 3 points)

This classifier implements a *weighted voting scheme*. All training points (not only the k nearest ones) cast a vote for the query point but their vote is weighted by a *Parzen window* (or *kernel function*) depending on the distance between the training samples $\underline{\mathbf{x}}^{(\alpha)}$ and the query point $\underline{\mathbf{x}}$. The Gaussian window function based on Euclidean norm $\|\cdot\|$ is:

$$\kappa(\underline{\mathbf{x}}, \underline{\mathbf{x}}^{(\alpha)}) = \exp\left(-\frac{1}{2\sigma_\kappa^2} \|\underline{\mathbf{x}} - \underline{\mathbf{x}}^{(\alpha)}\|^2 \right).$$

(a) (2 points) Plot the training patterns and the decision boundary (e.g. using a contour plot or a high-resolution image of equidistant query points) in input space for Gaussian window functions parameterized with the variances $\sigma_\kappa^2 = 0.5, 0.1$ and 0.01.

(b) (1 point) Add 60 new data points from a third class centered on $\underline{\tilde{\boldsymbol{\mu}}} = (0.5, 0.5)^\top$ with variance $\tilde{\sigma}^2 = 0.05$. Rerun the kNN and Parzen-window classification. Plot the classification boundaries as above and compare them with your previous results.

Exercise H7.4: RBF networks (homework, 4 points)

Similar to the Parzen window, RBF networks classify data according to a weighted vote, but the voting committee now consists of $K \ll p$ "representatives" instead of all p data points. These representatives do not have to be previously seen data points and can be "prototypes" $\underline{\mathbf{t}}_i \in \mathbb{R}^2$ derived from the training data via K-means clustering.

Construct an RBF network for binary classification – using the initial two-class data set from H7.1 and discarding the data points you added in H7.3b – as follows:

- Determine the K representatives $\underline{\mathbf{t}}_i$ via K*-means* clustering (you can implement the batch-algorithm described in the lecture or use an off-the-shelf implementation).
- For a given weight vector $\underline{\mathbf{w}} \in \mathbb{R}^{K+1}$, the predicted classification for a query point $\underline{\mathbf{x}}$ is:

$$y(\underline{\mathbf{x}}; \underline{\mathbf{w}}) = \mathrm{step}(\underline{\mathbf{w}}^\top \underline{\phi}(\underline{\mathbf{x}})),$$

[1] Python users can generate a grid of equidistant query points that cover the ranf of the input variables using numpy's meshgrid function, then color each grid point by how the classifier's prediction.

where $\underline{\phi}(\underline{\mathbf{x}}) := \begin{pmatrix} 1 \\ \phi_1(\underline{\mathbf{x}}) \\ \dots \\ \phi_K(\underline{\mathbf{x}}) \end{pmatrix}$ is a $(K+1)$-dimensional vector containing the bias and the basis function values $\phi_i(\underline{\mathbf{x}}) = \kappa(\underline{\mathbf{x}}, \underline{\mathbf{t}}_i)$ with κ from the previous exercise (Gaussian radial basis functions). Here we use the following step function

$$\text{step}(h) = \begin{cases} 1 & \text{for } h \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$$

to convert the network output $\underline{\mathbf{w}}^\top \underline{\phi}(\underline{\mathbf{x}})$ (regressed to labels 0 or 1) to a class prediction.

- Determine the weight vector as: $\underline{\mathbf{w}} = \left(\underline{\mathbf{\Phi}}\,\underline{\mathbf{\Phi}}^\top\right)^{-1} \underline{\mathbf{\Phi}}\,\underline{\mathbf{y}}_{\text{True}}^\top$
  where $\underline{\mathbf{y}}_{\text{True}} := (y_T^{(1)}, \dots, y_T^{(p)}) \in \mathbb{R}^{1,p}$ is the vector of target values and

$$\underline{\mathbf{\Phi}} := \left(\underline{\phi}(\underline{\mathbf{x}}^{(1)}), \dots, \underline{\phi}(\underline{\mathbf{x}}^{(p)})\right) \in \mathbb{R}^{K+1,p}$$

(a) (2 points) Plot the decision boundaries together with the training patterns and locations of the representatives for $K \in \{2, 3, 4\}$. Do this for two different (reasonable[2]) kernel widths σ_κ of the radial basis functions ϕ_i, yielding a total of six plots.

(b) (2 points) We would like to visualize how the non-linearly separable data appears in the transformed feature space, which the classifier operates on:

Construct a new RBF-network with 2 RBFs and fix the centers to $\underline{\mathbf{t}}_1 = (0,0)^\top$ and $\underline{\mathbf{t}}_2 = (1,1)^\top$ (i.e., skip K-means clustering).
For $\sigma_\kappa = 0.45$, produce a scatter plot of the data in the space of RBF-activations, i.e. for each data point $\underline{\mathbf{x}}^{(\alpha)}$, plot $\phi_1(\underline{\mathbf{x}}^{(\alpha)})$ vs. $\phi_2(\underline{\mathbf{x}}^{(\alpha)})$, indicate their class-assignment y_T by coloring the points accordingly.
Plot also the predicted labels after training in a similar second plot.

Feel free to reduce the data-variance σ (e.g. to 0.2) to make the cluster-structure more prominent.

Total 10 points.

[2] You have information to make an educated guess for the value of the kernel width. The second value should be chosen to reveal a qualitative effect of a lower or larger width on the predcitions.