Classification

Exercise T6.1: Radial basis function networks (tutorial)

(a) Describe and discuss the *general architecture* of an RBF-network. Would more than two layers improve the performance?

(b) Describe and discuss the *two-step learning procedure* for RBF-networks with k basis functions. Derive the analytical solution of the output-weights $\boldsymbol{w}$ for the cost function E^T, i.e.

$$E^T = \frac{1}{2p} \sum_{\alpha=1}^{p} \Big(y_T^{(\alpha)} - \sum_{j=0}^{k} w_j \phi_j(\boldsymbol{x}^{(\alpha)}) \Big)^2 .$$

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

Exercise T6.2: Multi-class classification (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*?

(c) Describe and discuss the multi-class predictor presented in the lecture. How is validation performed in this case? How can you predict the class of a previously unseen sample?

Exercise H6.1: Training data (homework, 1 point)

Create a sample of $P = 120$ training patterns $\{\underline{\mathbf{x}}_\alpha, t_\alpha\}$, $\alpha = 1, \ldots, P$. 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 two conditional distributions:

$$\begin{aligned} p_1 &:= p(\underline{\mathbf{x}}|\mathcal{C}_1) = \tfrac{1}{2}[\mathcal{N}(\underline{\mathbf{x}}|\underline{\mu}_1, \sigma^2) + \mathcal{N}(\underline{\mathbf{x}}|\underline{\mu}_2, \sigma^2)] \,, \\ p_2 &:= p(\underline{\mathbf{x}}|\mathcal{C}_2) = \tfrac{1}{2}[\mathcal{N}(\underline{\mathbf{x}}|\underline{\mu}_3, \sigma^2) + \mathcal{N}(\underline{\mathbf{x}}|\underline{\mu}_4, \sigma^2)] \,, \end{aligned}$$

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

- The corresponding target values $t_\alpha \in \{-1, 1\}$ describe the assignment to the two classes $\mathcal{C}_1, \mathcal{C}_2$ and indicate from which distribution (p_1 vs. p_2) 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 t_α.

Exercise H6.2: 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. Thus the *electoral committee* is selected from the training patterns $\{\underline{\mathbf{x}}_\alpha, t_\alpha\},\ \alpha = 1, \ldots, P$ according their Euclidean distance to the query point. The predicted class is determined by the target values 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 equidistant query points) in input space for $k = 1, 3, 5$.

Exercise H6.3: "Parzen window" classifier (homework, 3 points)

This classifier implements a *weighted voting scheme*. All training points (not only the k nearest ones) make 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 query point $\underline{\mathbf{x}}$. The Gaussian window function based on Euclidean norm $\|\cdot\|$ is:

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

(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) Rerun kNN and Parzen-window classification after adding 60 more data points from a third class centered on $\underline{\mu}_3 = (0.5, 0.5)^\top$ with variance $\tilde{\sigma}^2 = 0.05$. Plot the classification boundaries as above and compare them with your previous results.

Exercise H6.4: 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 $k < P$ "representatives", which parametrize the RBFs. These representatives do not have to be previously seen data points and can be "prototypes" $\underline{\mathbf{v}}_j \in \mathbb{R}^2$ derived from the training data via k-means clustering. Construct a RBF-net as follows:

- Determine the k representatives $\underline{\mathbf{v}}_j$ via *k-means* clustering (you can implement the online-algorithm described in the lecture notes or use available packages).
- For a given weight vector $\underline{\mathbf{w}}$, the predicted classification for a query point $\underline{\mathbf{x}}$ is:

$$y(\underline{\mathbf{x}}) = \operatorname{sign}(\underline{\mathbf{w}}^\top \underline{\phi}(\underline{\mathbf{x}})),$$

  where $\underline{\phi}(\underline{\mathbf{x}})$ is a $(k+1) \times 1$ vector containing the bias and the basis function values $\phi_i(\underline{\mathbf{x}})$.

- Determine the weight vector as: $\underline{\mathbf{w}} = \left(\underline{\mathbf{\Phi}}\underline{\mathbf{\Phi}}^\top\right)^{-1}\underline{\mathbf{\Phi}}\underline{\mathbf{t}} \equiv \underline{\mathbf{\Phi}}^{\dagger\top}\underline{\mathbf{t}}$ where $\underline{\mathbf{t}} \in \mathbb{R}^P$ is the vector of target values and $\underline{\mathbf{\Phi}} \in \mathbb{R}^{k+1\times P}$ is the $k+1 \times P$ *design matrix* with

$$\Phi_{0\alpha} := 1\,, \quad \text{and} \quad \Phi_{j\alpha} := \phi_j(\underline{\mathbf{x}}_\alpha) \equiv \kappa(\underline{\mathbf{x}}_\alpha, \underline{\mathbf{v}}_j), \quad j = 1, \ldots, k; \;\; \alpha = 1, \ldots, P\,.$$

  You can use predefined functions to calculate the pseudo-inverse $\underline{\mathbf{\Phi}}^\dagger$ (e.g. `linalg.pinv` in Python or `pinv` in Matlab).

(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) kernel widths σ_κ of the radial basis functions ϕ_j, yielding a total of six plots.

(b) (2 points) Construct a RBF-network with 2 RBFs and set the centers to $\underline{\mu}_1 = (0, 0)$ and $\underline{\mu}_2 = (1, 1)$. For $\sigma_\kappa = 0.45$, make a scatter plot of the data in the space of RBF-activations, i.e. for each data point α plot $\phi_1(x_\alpha)$ vs. $\phi_2(x_\alpha)$ and indicate their class-assignment via their color. 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.