Course: Machine learning

By: Pavlo Mozharovskyi

Tutorial to Lecture 5 (part `R`): Perceptron, neural network, and the back-propagation algorithm

Task 1: Rosenblatt's perceptron.

(a) Visualize two variables *sepal length* and *sepal width* of the two classes, *setosa* and *versicolor*, of the *iris* data set. You can obtain the data set by simply typing `data(iris)`. Do they seem linearly separable?

(b) Program the original algorithm of the Rosenblatt's perceptron. For this, create functions `perceptron.train` (which takes training sample, learning rate and initial values of the normal vector and of the intercept as input arguments; you can additionally provide upper bound on the number of iterations) and `perceptron.classify` (which takes new observations, the normal vector and the intercept as input arguments).

(c) Visualize the perceptron rule trained on the data from part (a): On a 2D-plot show training observations and fill the areas of different class assignment with different colors. You can use the provided function `filled.contour2`.

(d) The exclusive logical disjunction or eXclusive logical OR, usually shortened and known as XOR, can be described by the following table:

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Let us call *noisy XOR* data generated according to the following scheme ($n = 100$ being the total number of points): Let a set consisting of $n/4$ points generated from $\mathcal{N}(\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right], \left[\begin{smallmatrix}0.01 & 0\\0 & 0.01\end{smallmatrix}\right])$ and of $n/4$ points generated from $\mathcal{N}(\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right], \left[\begin{smallmatrix}0.01 & 0\\0 & 0.01\end{smallmatrix}\right])$ have label 0 and a set consisting of $n/4$ points generated from $\mathcal{N}(\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right], \left[\begin{smallmatrix}0.01 & 0\\0 & 0.01\end{smallmatrix}\right])$ and of $n/4$ points generated from $\mathcal{N}(\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right], \left[\begin{smallmatrix}0.01 & 0\\0 & 0.01\end{smallmatrix}\right])$ have label 1. Visualize this data set. Are the two classes linearly separable?

(e) Create functions `perceptron2.train` and `perceptron2.classify` which call functions `perceptron.train` and `perceptron.classify`, respectively, feeding them with second-order polynomial extension of the data. For example, a two-dimensional input vector $\boldsymbol{x} = (x_1, x_2)^T$ should be preliminary transformed into $\tilde{\boldsymbol{x}} = (x_1, x_2, x_1 \cdot x_2, x_1^2, x_2^2)^T$. To save time you can restrict here to two-dimensional data only. We will call this a generalized perceptron.

(f) Visualize the generalized perceptron rule trained on the data from part (d): On a 2D-plot show training observations and fill the areas of different class assignment with different colors.

Task 2: Classification with a multilayer perceptron.

(a) Generate data from distributional setting 1 from Table 1 (Appendix of Tutorial 2) with a training sample containing 200 observations (100 observations from each class). What is the optimal classifier for this setting? Train a single neuron using the back-propagation algorithm (you can use function `neuralnet` from R-package `neuralnet`). Visualize the structure of this neuron (you can use the generic function `plot` or function `plotnet` from R-package `NeuralNetTools`). Explain the obtained plot. Visualize the classification rule of the trained neuron: On a 2D-plot show training observations and fill the areas of different class assignment with different colors (you can use functions `compute` and `filled.contour2`).

(b) Train the linear discriminant analysis (LDA) classifier on the same data (you can use function `lda` from R-package `MASS`). For both neuron from (a) and the trained LDA compute (and print) the (unit) normal vector of the separating line. Further compute the angle between these two vectors in degrees (and print it).

(c) Generate data from distributional setting 2 from Table 1 (Appendix of Tutorial 2) with a training sample containing 200 observations (100 observations from each class). What is the optimal classifier for this setting? Train a two-layer perceptron using the back-propagation algorithm. Visualize its structure and explain the obtained plot. Further, visualize its classification rule: On a 2D-plot show training observations and fill the areas of different class assignment with different colors.

(d) Train the quadratic discriminant analysis (QDA) classifier on the data from (c) and plot its separating line on the plot from (c) (you can use function `contour`). Are the two rules similar? Repeat training phase in (c) to obtain a rule resembling this by QDA until you are satisfied with the result.

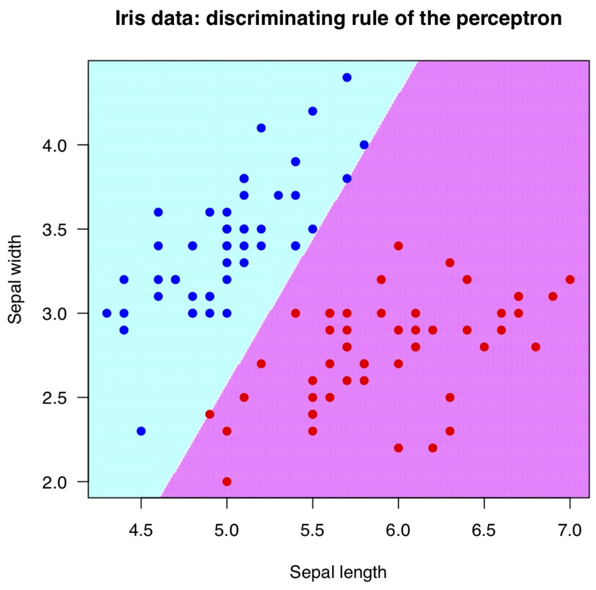
Task 3: Reconstruction of functional dependencies.

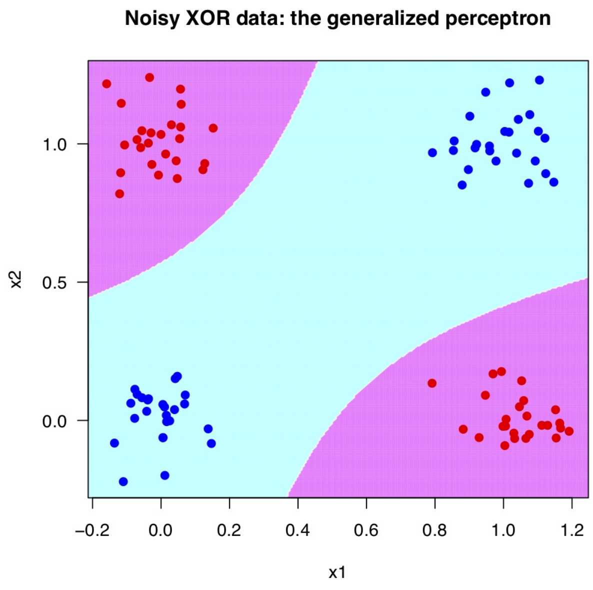
(a) Generate n $(= 100)$ inputs $\{x_1, ..., x_n\}$ uniformly distributed on the interval $[0, 2\pi]$ and their corresponding outputs being $y_i = \sin(x_i) + \epsilon_i$ with $\epsilon_i \sim \mathcal{N}(0, 0.1^2)$ for $i = 1, ..., n$. Draw the points $(x_i, y_i)^\top$, $i = 1, ..., n$ on a plot with limits $[0, 6\pi]$. Superpose the $y = \sin(x)$ function on this plot. Train a multilayer perceptron using the back-propagation algorithm and taking these n points as training data to approximate the sin function. Visualize the network's structure. Superpose the predicted function for each (visible) value of the abscissa on the existing plot. Is the approximation/prediction reliable? Retrain the neural network to predict as good as possible.
Hint: use a radial-basis activation function, *e.g.*, $\phi(v) = e^{-v^2}$.

(b) Generate n $(= 100)$ bivariate inputs $\{(x_{11}, x_{12})^\top, (x_{21}, x_{22})^\top, ..., (x_{n1}, x_{n2})^\top\}$ uniformly distributed on $[0, 1]^2$ and their corresponding outputs being $y_i = \cos(\frac{\sqrt{x_{i1}^2+x_{i2}^2}}{\sqrt{2}}\pi) + \epsilon_i$ with $\epsilon_i \sim \mathcal{N}(0, 0.05^2)$ for $i = 1, ..., n$. Visualize these n points $(x_{i1}, x_{i2}, y_i)^\top$, $i = 1, ..., n$ on a 3D-plot. You can use function `cloud` from `R`-package `lattice`. Train a multilayer perceptron using the back-propagation algorithm and taking these n points as training data to approximate the function $\cos(\frac{\sqrt{x_{i1}^2+x_{i2}^2}}{\sqrt{2}}\pi)$. Visualize the network's structure. Visualize the obtained surface on $[0, 1]^2$. You can use function `wireframe` from `R`-package `lattice`. Is the approximation reliable?

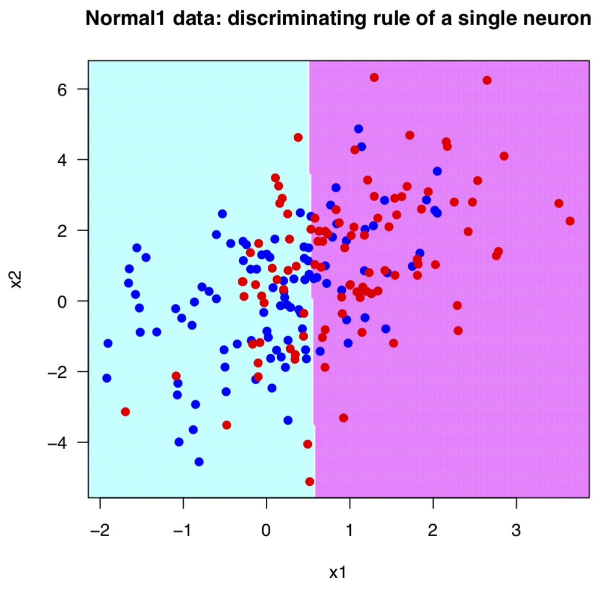
Appendix: Examples of produced figures

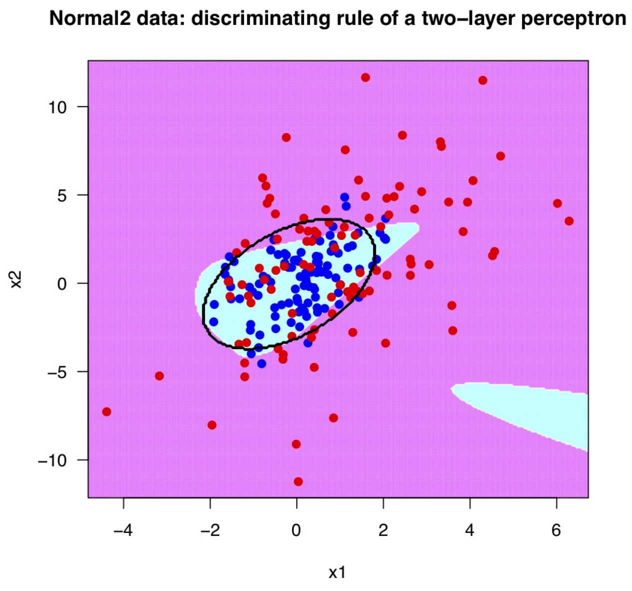
1. Task 1: Rosenblatt's perceptron.



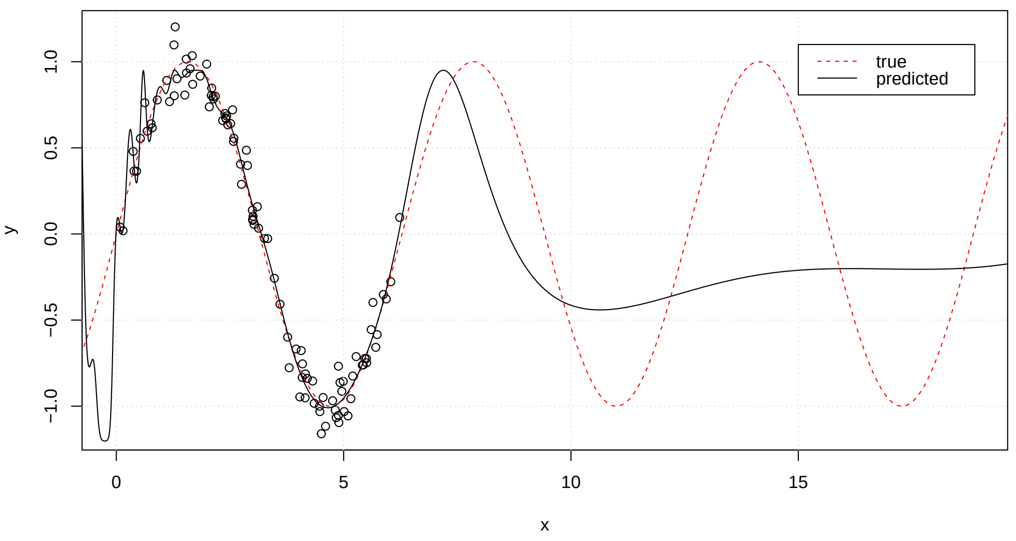


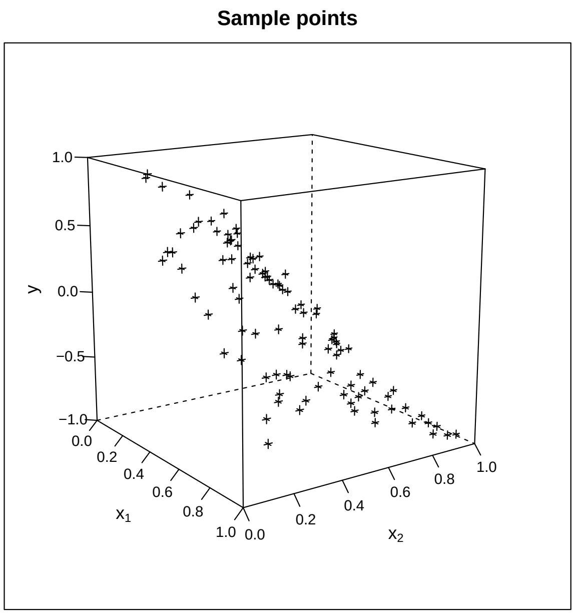
2. Task 2: Classification with a multilayer perceptron.

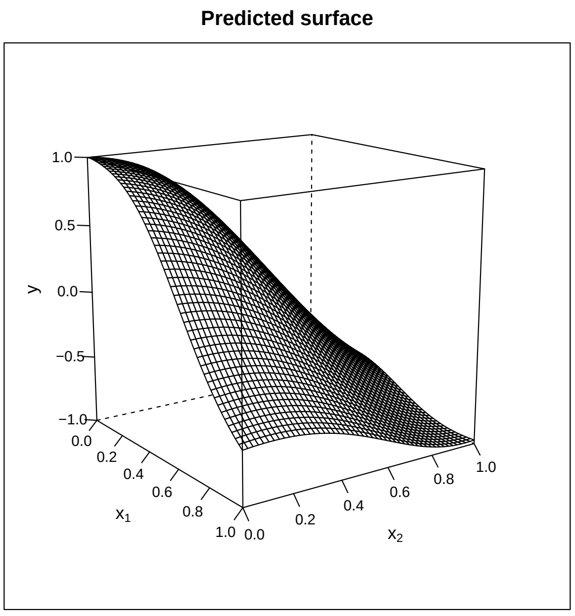




3. Task 3: Reconstruction of functional dependencies.


y
x
true
predicted


Sample points
y
x1
x2


Predicted surface
y
x1
x2