Multilayer Perceptrons and Backpropagation Algorithm

Exercise T3.1: Cost functions (tutorial)

(a) What effect will the choice of error measure (particularly *quadratic or linear*) produce?

(b) Outline the relation between the quadratic error function and the Gaussian conditional distribution for the labels.

(c) Derive a suitable error function (*cross entropy*) for the following case: the output of a neural network is interpreted as the probability that the input belongs to the first of two classes.

(d) Summarize the error measures and output layers for regression and classification.

Exercise T3.2: Parameter optimization (tutorial)

(a) Recap MLP architecture, outline gradient descent, and derive the back propagation algorithm (backprop) for a MLP with L layers.

(b) Discuss the consequence of parameter space symmetries: (i) permutation of neuron indices within a layer, (ii) reversal of signs across consecutive layers.

Exercise H3.1: Binary Classification (homework, 3 points)

For binary targets $y_T^{(\alpha)} \in \{0, 1\}$ the network output $y(\underline{\mathbf{x}}; \underline{\mathbf{w}}) \in (0, 1)$ can be interpreted as a probability $P(y = 1|\underline{\mathbf{x}}; \underline{\mathbf{w}})$. A suitable error function for this problem is:

$$E^T = \frac{1}{p} \sum_{\alpha=1}^{p} e^{(\alpha)}$$

with

$$e^{(\alpha)} = -\left[y_T^{(\alpha)} \ln y(\underline{\mathbf{x}}^{(\alpha)}; \underline{\mathbf{w}}) + (1 - y_T^{(\alpha)}) \ln \left(1 - y(\underline{\mathbf{x}}^{(\alpha)}; \underline{\mathbf{w}}) \right) \right].$$

(a) (1 point) Show that

$$\frac{\partial e^{(\alpha)}}{\partial y(\underline{\mathbf{x}}^{(\alpha)}; \underline{\mathbf{w}})} = \frac{y(\underline{\mathbf{x}}^{(\alpha)}; \underline{\mathbf{w}}) - y_T^{(\alpha)}}{y(\underline{\mathbf{x}}^{(\alpha)}; \underline{\mathbf{w}}) \left(1 - y(\underline{\mathbf{x}}^{(\alpha)}; \underline{\mathbf{w}}) \right)}$$

(b) (1 point) Consider an MLP with one hidden layer. The nonlinear transfer function for the output neuron $(i = 1, v = 2)$ is assumed to be

$$f(h_1^2) = \frac{1}{1 + \exp(-h_1^2)},$$

where h_1^2 is the total input[1] of the output neuron. Show that its derivative can be expressed as

$$f'(h_1^2) = f(h_1^2)(1 - f(h_1^2)).$$

[1] The total input of a neuron is sometimes referred to as a *logit*

(c) (1 point) Using the results from a) and b), show that the gradient of the error function $e^{(\alpha)}$ with respect to the weight w_{1j}^{21} between the the single output neuron $(i = 1, v = 2)$ and neuron j of the hidden layer $(j > 0, v = 1)$ is

$$\frac{\partial e^{(\alpha)}}{\partial w_{1j}^{21}} = \left(y(\underline{\mathbf{x}}^{(\alpha)}; \underline{\mathbf{w}}) - y_T^{(\alpha)} \right) f(h_j^1).$$

Exercise H3.2: MLP Regression (homework, 7 points)

The task is to implement an MLP with one hidden layer and apply the backpropagation algorithm to learn its parameters for a regression task.

Training Data: The file `RegressionData.txt` from the ISIS platform contains a small training dataset $\{x^{(\alpha)}, y_T^{(\alpha)}\}$, $\alpha = 1, \dots, p$ with $p = 10$. The input values $\{x^{(\alpha)}\}$ in the first column are random numbers drawn from a uniform distribution over the interval $[0, 1]$. The target values $\{y_T^{(\alpha)}\}$ were generated using the function $\sin\left(2\pi x^{(\alpha)}\right)$ and Gaussian noise with standard deviation $\sigma = 0.25$ was added.

(A) **Initialization:**

1. Construct the MLP using a single hidden layer with 3 hidden nodes ($N_1 = 3$) and an output layer with a single output neuron ($N_L = N_2 = 1$).
2. Use the $\tanh$ transfer function for the hidden neurons and the linear transfer function (i.e. the identity) for the output neuron.
3. Set the weights and biases to random values from the interval $[-0.5, 0.5]$.

(B) **Iterative learning:**

1. For each input value $x^{(\alpha)}$ of the training set, do:
   (a) **Forward Propagation:** Calculate the activity of the hidden neurons and the output neuron.
   (b) Compute the **output error** $e^{(\alpha)}$ using the quadratic error cost function.
   (c) **Backpropagation:** Calculate the "local errors" δ_i^v for the output and the hidden layer for each training point.
   (d) Calculate the gradient of the error function w.r.t. the first and second layer weights $w_{ij}^{1\,0}$ and $w_{ij}^{2\,1}$ respectively [2].
2. Calculate the batch gradient in order to obtain the direction of the weight updates:

$$\Delta w_{ij}^{v'v} = -\frac{\partial E_{[\mathbf{w}]}^T}{\partial w_{ij}^{v'v}} = -\frac{1}{p} \sum_{\alpha=1}^{p} \frac{\partial e_{[\mathbf{w}]}^{(\alpha)}}{\partial w_{ij}^{v'v}}$$

   where $j = 0, \dots, N_v$ and $i = 1, \dots, N_{v'}$

[2] The weights in the second layer are those connecting the hidden neurons with the output node. Since we only have one output neuron, $w_{ij}^{2\,1}$ is effectively $w_{1j}^{2\,1}$

3. **Weight update:** Use gradient descent with a fixed learning rate $\eta = 0.5$ to update the weights in each iteration according to

$$\underline{\mathbf{w}}^{(t+1)} = \underline{\mathbf{w}}^{(t)} + \eta \Delta \underline{\mathbf{w}}^{(t)}$$

(C) **Stopping criterion**:
Stop the iterative weight updates if the error E^T has converged, i.e. $|\Delta E^T|/E^T$ has fallen below some small value (e.g. 10^{-5}) or a maximum number of iterations $t_{max} = 3000$ has been reached.

Devliverables:

(a) (2 point) Plot the error E^T over the iterations.

(b) (1 point) For the final network, plot the output of hidden units for all inputs.

(c) (1 point) Plot the output values over the input space (i.e. the input-output function of the network) together with the training dataset.

(d) (2 point) Plot (a)–(c) *twice* (i.e., for different initial conditions) next to each other and discuss: is there a difference, and if so, why?

(e) (1 point) What might have been the motivation for using a quadratic error function here?

Total 10 points.