Statistical Methods for Machine Learning
Assignment 2: Basic Learning Algorithms

Philip Pickering
pgpick@gmx.at

Marco Eilers
eilers.marco@googlemail.com

Thomas Bracht Laumann Jespersen
ntl316@alumni.ku.dk

1 Neural Networks

1.1 Neural Network implementation

Implement a multi-layer neural network with linear output neuron and a single hidden layer with non-linear neuron. All neurons should have bias (offset) parameters.

To find the derivative of the activation function:

$$\sigma(u) = \frac{|u|}{1+|u|}$$

we apply the quotient rule for differantiation:

$$\frac{d}{du}\frac{f(x)}{g(x)} = \frac{g(x)f'(x) - g'(x)f(x)}{[g(x)]^2}$$

where, in our case $f(u) = |u|$ and $g(u) = 1 + |u|$.

$$\frac{d}{du}\left(\frac{|u|}{1+|u|}\right) = \frac{(1+|u|)\cdot 1 - (0+1)|u|}{(1+|u|)^2} \tag{1}$$

$$= \frac{1+|u|-|u|}{(1+|u|)^2} \tag{2}$$

$$= \frac{1}{(1+|u|)^2} \tag{3}$$

Implement backpropagation to compute gradient of error with respect to the network parameters.

1.2 Neural Network training

Fig. 1 plots the $\frac{\sin x}{x}$ on the range $[-10, 10]$ along with the predictions of our trained NN model.

Figure 1: Plot of $\frac{\sin x}{x}$ width neural network predictions for the same range.

2 Support Vector Machines

For this part of the assignment we chose to use the LIBSVM software.

2.1 Model Selection

Description (we normalized the data, then used the builtin function of libsvm, tried these values for gamma: [])

We did grid search using the following values of $\gamma : \{0.0001, 0.001, 0.01, 0.1, 1, 10, 100\}$. This choice is based on what?

LIBSVM has built-in functionality to perform n-fold cross validation given a command line option. To perform model selection we iterate through all combinations of C and γ and call a function called `crossval`, which invokes LIBSVM to perform a 5-fold cross validation on the current values of C and γ. When performing n-fold cross validation, LIBSVM returns the accuracy, which we use to keep track of the configuration that gives the highest accuracy.

γ \ C	0.1	1	10	100	1000	10000
0.0001	55%	55%	55%	55%	55%	55%
0.001	55%	55%	55%	55%	55%	65%
0.01	55%	55%	56%	57%	65%	98%
0.1	60%	60%	59%	63%	98%	95%
1	57%	58%	59%	92%	97%	94%
10	51%	54%	67%	88%	92%	91%
100	52%	47%	76%	76%	77%	77%

Table 1: Table of all results for model selection using grid-search for `knollC-train100`.

γ \ C	0.1	1	10	100	1000	10000
0.0001	55%	55%	55%	55%	55%	55%
0.001	55%	55%	55%	55%	55%	65%
0.01	55%	55%	56%	57%	65%	98%
0.1	60%	60%	59%	63%	98%	95%
1	57%	58%	59%	92%	97%	94%
10	51%	54%	67%	88%	92%	91%
100	52%	47%	76%	76%	77%	77%

Table 2: Table of all results for model selection using grid-search for `knollC-train200`.

Applied to the testdata, this gives the results in table 1 to 3. The resulting optimal parameter configurations are shown in table 4.

2.2 Inspecting the kernel expansion

2.2.1 Visualization

Fig. 2 shows the plot of the `knollC-train200` data set, in which the support vectors are circled. The free support vectors are circled in black, and bounded are circled in green. There are 87 bounded support vectors, and just six free for a total of 93 support vectors.

γ \ C	0.1	1	10	100	1000	10000
0.0001	55%	55%	55%	55%	55%	55%
0.001	55%	55%	55%	55%	55%	65%
0.01	55%	55%	56%	57%	65%	98%
0.1	60%	60%	59%	63%	98%	95%
1	57%	58%	59%	92%	97%	94%
10	51%	54%	67%	88%	92%	91%
100	52%	47%	76%	76%	77%	77%

Table 3: Table of all results for model selection using grid-search for `knollC-train400`.

	C	γ	Acc.
`knollC-train100`	1000	0.1	98%
`knollC-train200`	1000	0.1	97.5%
`knollC-train400`	100	1	97.5%

Table 4: Table of results for model selection using grid-search showing the optimal values for C and γ.

Model \ Data	`knollC-train100`	`knollC-train200`	`knollC-train400`	`knollC-test`
`knollC-train100`	98%	97.5%	96.75%	97%
`knollC-train200`	98%	98%	97.5%	98%
`knollC-train400`	99%	98.5%	97.25%	98%

Table 5: Percentage of correct predictions when applying the model from each training data set to the training and test data.

Figure 2: `knollC-train200` data set with free support vectors (circles) and bounded support vectors (squares).

2.2.2 Effect of the regularization parameter

The file `regularization.m` performs the outlined procedure, by first training the SVM model using the values for C and γ found during model selection. Then it trains to other models, one in which C is multiplied by a hundred and one in which we divide C by 100.

The most notable change is in the number of support vectors. There's a total of 93 support vectors for the "original" value of C—87 of which are bounded. When C is a hundred times larger, the number of support vectors drop to just 19, all of which are free. Conversely, when dividing C by a hundred we get an increase in the number of support vectors to 199, but again all of them are free.

2.2.3 Scaling behaviour

Table of free and bounded

	bounded	free
`knollC-train100`	5	60
`knollC-train200`	6	87
`knollC-train400`	12	153

Table 6: Table of bounded and free support vectors for the three data sets.