Machine Learning
Winter Semester 2019/2020

Sheet #7 – Solution

Due:

Fred Hamprecht
Manuel Haußmann

1 Multilayer Perceptron (10 pt)

Throughout this work you will consider an MLP.

i) Encoding logic in an MLP. Assuming your input to the MLP is a 2 dimensional binary vector, $\mathbf{x} \in \{0,1\}^2$. How would you encode a logical AND (the output is equal to 1 iff[1] $x_i == 1 \;\; \forall i$), OR (the output is equal to 1 iff $x_i = 1$ for at least one x_i), and XOR (the output is equal to 1 iff $x_i = 1$ for exactly one i).

ii) Splitting of the feature space. Assuming you have a layer of M perceptrons (i.e. M lines). Assuming some regularity (i.e. no two decision boundaries are parallel and no three decision boundaries cross in a single point), in how many regions will they split the plane?

iii) An Example Split the feature space in Figure 1 optimally, and give the binary output vectors you get for each of the resulting regions. How would the corresponding region splitting a classification tree would produce (draw it once assuming axes aligned splits, and once assuming oblique splits).

Solution:

i) Throughout the output of a layer is given as $y = \texttt{step}(\mathbf{w}^T\mathbf{x} + b)$.

AND As we we already have a binary $\mathbf{x}$, we can set $\mathbf{w} = \mathbf{1}_2$ and $b = -1.5$. This gives the desired solution as $\mathbf{w}^T\mathbf{x} + b > 0$ iff the $x_i = 1 \;\; \forall i$.

OR Choosing $\mathbf{w} = \mathbf{1}_2$ and $b = -0.5$ gives the OR case.

XOR This is the interesting case as we can no longer solve it with a single layer. Make use of the fact that XOR(a,b) = AND(OR(a,b),NOT(AND(a,b))). The AND and OR cases are already given above, and we only require the NOT(AND(a,b)) (NAND). For that we use $\mathbf{w}_{\text{NAND}} = -\mathbf{1}_2$ and $b_{\text{NAND}} = 1.5$. This gives

$$\begin{aligned}
y^{(1)}_{\text{OR}} &= \texttt{step}(\mathbf{w}^{(1)}_{\text{OR}}\mathbf{x} + b^{(1)}_{\text{OR}}) \\
y^{(1)}_{\text{NAND}} &= \texttt{step}(\mathbf{w}^{(1)}_{\text{NAND}}\mathbf{x} + b^{(1)}_{\text{NAND}}) \\
\mathbf{y}^{(1)} &= [y^{(1)}_{\text{OR}}, y^{(1)}_{\text{NAND}}]^T \\
y^{(2)}_{\text{AND}} &= \texttt{step}(\mathbf{w}^{(2)}_{\text{AND}}\mathbf{y}^{(1)} + b^{(2)}_{\text{AND}}),
\end{aligned}$$

where the superscript indicates the layer.

ii) A single decision boundary splits the plane into two $R(1) = 2$. The M-th decision boundary cuts all $M-1$ decision boundaries and gives M new ones. $R(M) = R(M-1) + M$. This gives us in total $R(M) = 1 + \frac{M(M+1)}{2}$.[2]

iii) For the regions see `decision_boundaries.pdf`. Using the region names given there the binary output vectors for each region are summarized in Table 1.

[1] if and only if

[2] For a more general discussion of the number of regions in a deep neural network with ReLU activations see e.g. Montúfar et al., 2014 (https://arxiv.org/abs/1402.1869)

Table 1: The MLP Region output vectors

	A	B	C	D	E	F	G
1	0	0	1	0	0	1	1
2	0	0	0	1	1	1	1
3	1	0	0	1	0	0	1

2 Training of an MLP (12 pt)

So far we haven't discussed in detail how to train a multilayer perceptron (MLP). For that reason we will rely on `http://playground.tensorflow.org` to nevertheless gain some intuition on what happens throughout the training of a neural network and how the different pieces interact and work together. For each part submit a screenshot and a short discussion of what you observed.

i) Fitting a Neural Net. Consider the spiral data set and the first two features (X_1, X_2). Come up with an architecture that can learn to classify the pattern well. You are free to use any number layers/neurons[3]/activation functions/regularization/...

ii) Exploring Regularization. Pick the largest network size (i.e. 6 layers of 8 neurons each) and one of the data sets. Train it first without regularization, observing the behavior. Retrain it with L1 and L2 regularization and observe how the weight structure changes. What kind of behavior do you expect and does it fit with what you observe?

iii) Breaking Things. As discussed in the lecture, a net with enough parameters can fit any kind of pattern, even if there is none. Try to replicate this observation. Have your net learn a pattern "perfectly" (i.e. very low training error), but without having predictive power (i.e. the test error stays larger than random (which would be 0.5)). *Hint: If you consider the spiral data set with the minimal amount of training data and the maximum amount of noise, you get a collection of points with most structure removed.*

Solution: See the accompanying plots, i.e. `ex2*.png` which are combined in `ex07-solutionplots.zip`.

[3] A neuron here is what we in the lecture talked about as a perceptron.