Artificial Neural Networks (Gerstner). Solutions for week 3

Statistical Approaches for Deep Networks

Exercise 1. Maximum Likelihood solution

You have observed P data points, $\{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^P\}$ where $\boldsymbol{x}^\mu \in \mathcal{R}^N$.

You have reason to believe that the data set might come from a radially symmetric Gaussian distribution

$$p(\boldsymbol{x}|\mathrm{x}_c, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}^N} \exp\left[-\frac{(\boldsymbol{x} - \boldsymbol{x}_c)^\top (\boldsymbol{x} - \boldsymbol{x}_c)}{2\sigma^2}\right] \tag{1}$$

where $\boldsymbol{x}_c$ is the center Gaussian and σ its width (standard deviation).

a. In-class-exercise: Calculate the value of $\boldsymbol{x}_c$ for which the likelihood that the data could have been generated by a Gaussian is maximal.

b. At home: Repeat the same calculation for the width σ.

Solution:

a. The arg max of the joint *log* likelihood of all P points is more convenient to compute and gives the same result, because the log function is strictly monotonically increasing. We have

$$\begin{aligned}
p(\{\mathbf{x}^\mu\}_1^P|\mathbf{x}_c, \sigma) &= \prod_{i=1}^{P} \frac{1}{\sqrt{2\pi\sigma^2}^N} \exp\left[-\frac{(\mathbf{x}_i - \mathbf{x}_c)^\top (\mathbf{x_i} - \mathbf{x}_c)}{2\sigma^2}\right] \\
\ln p(\{\mathbf{x}^\mu\}_1^P|\mathbf{x}_c, \sigma) &= \sum_{i=1}^{P} \left[\ln\left[\frac{1}{\sqrt{2\pi\sigma^2}^N}\right] + \left[-\frac{(\mathbf{x}_i - \mathbf{x}_c)^\top (\mathbf{x}_i - \mathbf{x}_c)}{2\sigma^2}\right]\right] \\
\frac{\partial \ln p(\{\mathbf{x}^\mu\}_1^P|\mathbf{x}_c, \sigma)}{\partial \mathbf{x}_c} &= \sum_{i=1}^{P} \frac{(\mathbf{x}_i - \mathbf{x}_c)}{\sigma^2} \\
0 &= \sum_{i=1}^{P} (\mathbf{x}_i - \mathbf{x}_c) \\
\mathbf{x}_c &= \frac{1}{P} \sum_{i=1}^{P} \mathbf{x}_i
\end{aligned}$$

which is the arithmetic mean.

b.

$$\begin{aligned}
\ln p(\{\mathbf{x}^{\mu}\}_1^P|\mathbf{x}_c,\sigma) &= \sum_{i=1}^{P}\left[\ln\left[\frac{1}{\sqrt{2\pi\sigma^2}^N}\right]+\left[-\frac{(\mathbf{x}_i-\mathbf{x}_c)^\top(\mathbf{x}_i-\mathbf{x}_c)}{2\sigma^2}\right]\right]\\
\ln p(\{\mathbf{x}^{\mu}\}_1^P|\mathbf{x}_c,\sigma) &= \sum_{i=1}^{P}\left[-\frac{1}{2}N\ln(2\pi)-N\ln\sigma-\left[\frac{(\mathbf{x}_i-\mathbf{x}_c)^\top(\mathbf{x}_i-\mathbf{x}_c)}{2\sigma^2}\right]\right]\\
\frac{\partial\ln p(\{\mathbf{x}^{\mu}\}_1^P|\mathbf{x}_c,\sigma)}{\partial\sigma} &= \sum_{i=1}^{P}\left[\frac{-N}{\sigma}+\frac{(\mathbf{x}_i-\mathbf{x}_c)^\top(\mathbf{x}_i-\mathbf{x}_c)}{\sigma^3}\right]\\
0 &= \sum_{i=1}^{P}\left[-N+\frac{(\mathbf{x}_i-\mathbf{x}_c)^\top(\mathbf{x}_i-\mathbf{x}_c)}{\sigma^2}\right]\\
0 &= -NP+\frac{1}{\sigma^2}\sum_{i=1}^{P}(\mathbf{x}_i-\mathbf{x}_c)^\top(\mathbf{x}_i-\mathbf{x}_c)\\
\sigma &= \left(\frac{1}{NP}\sum_{i=1}^{P}(\mathbf{x}_i-\mathbf{x}_c)^\top(\mathbf{x}_i-\mathbf{x}_c)\right)^{\frac{1}{2}}
\end{aligned}$$

Exercise 2. Logistic/Sigmoidal unit

In class it was shown that the drive a for the sigmoidal unit in a single-class scenario is given by the log of the probability ratio $a = \ln[p(\boldsymbol{x}, C_1)/p(\boldsymbol{x}, \bar{C}_1)]$ where the bar indicates “not in class C_1”.

In some textbooks, however, it is argued that the drive a is rather $a = \ln[p(C_1|\boldsymbol{x})/p(\bar{C}_1|\boldsymbol{x})]$.

a. Are the two expressions for a the same or not?

b. Define $b = \ln[p(\boldsymbol{x}|C_1)/p(\boldsymbol{x}|\bar{C}_1)]$. Is $b = a$? Always true, never, or under some condition?

Solution:

a.

$$\begin{aligned}
a &= \ln\frac{p(\mathbf{x}, C_1)}{p(\mathbf{x}, \bar{C}_1)}\\
&= \ln\frac{p(C_1|\mathbf{x})p(\mathbf{x})}{p(\bar{C}_1|\mathbf{x})p(\mathbf{x})}\\
&= \ln\frac{p(C_1|\mathbf{x})}{p(\bar{C}_1|\mathbf{x})}
\end{aligned}$$

so the two expressions are the same.

b.

$$\begin{aligned}
a &= \ln\frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|\bar{C}_1)p(\bar{C}_1)}\\
&= b+\ln\frac{p(C_1)}{p(\bar{C}_1)},
\end{aligned}$$

so $a = b$ if $p(C_1) = p(\bar{C}_1)$, i.e. if it is equally likely for a data sample to come from the class or to not come from the class. In a single–class case, $p(\bar{C}_1) = 1 - p(C_1)$, so this would occur if $p(C_1) = p(\bar{C}_1) = 0.5$.

Exercise 3. Softmax function

In class we have defined the softmax output for unit k as

$$\hat{y}_k = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \tag{2}$$

The following considerations are important in a practical implementation of the softmax function in order to avoid overflow and underflow events.

a. Show that the output does not change if you shift, in all components, the activations by the same constant c. In other words, you may work with $a_j + c$ instead of a_j, for all j simultaneously.

b. Why is it useful to choose $c = -\max_j[a_j]$?

c. Check on your calculator that $\exp(3) \approx 20$. Your friend argues that, by subtracting the constant defined in part b, you can neglect all outputs $a_j - \max_j[a_j] < -10$. Is she correct?

d. You have a network with $K = 4$ output units in layer n. We have a multi-class problem with 1–hot coding. For each input $\boldsymbol{x}^\mu$ only one of the $K = 4$ output units has target $t_i^\mu = 1$ and all other target values are zero: $t_k^\mu = 0$ for $k \neq i$. The cross-entropy error function is defined as

$$E(\boldsymbol{w}) = -\sum_\mu \sum_{k=1}^{4} t_k^\mu \ln \hat{y}_k^\mu \tag{3}$$

where the sums run over all output units and all patterns. Assume that the output $\hat{y}_k$ is given by the softmax function defined at the beginning of the exercise, with a drive $a_k = \sum_j w_{kj}^{(n)} x_j^{(n-1)}$.

Perform stochastic gradient descent. To do so, consider one specific pattern, e.g., $\mu = 12$. Give the update formula for weight $w_{34}^{(n)}$ after the presentation of pattern $\mu = 12$. Hint: exploit that in the 1-hot coding $\sum_k t_k^\mu = 1$ for all μ.

e. The target output for pattern $\mu = 12$ is $t_1^{12} = 1$ and zero for the other output components. The drives for the output units are $a_1 = 100$; $a_2 = 97$; $a_3 = 92$; $a_4 = 14$. Without using a calculator, which output unit has the largest absolute output error $|\delta| = |dE/da|$? Hint: use the results of (a) - (d).

f. Show that for the case of two classes and 1-hot coding (1 out of K), the multi-class cross-entropy in (d) reduces to the single–class formula:

$$E(\boldsymbol{w}) = -\sum_\mu [t^\mu ln\hat{y}^\mu + (1 - t^\mu) ln(1 - \hat{y}^\mu)] \tag{4}$$

and with the softmax replaced by the sigmoidal.

Hint: Start with the softmax definition for $K = 2$ classes and exploit that $\hat{y}_1 + \hat{y}_2 = 1$.

Solution:

a.

$$\begin{aligned}
\hat{y}_k &= \frac{\exp(a_k + c)}{\sum_j \exp(a_j + c)} \\
&= \frac{\exp(a_k)\exp(c)}{\exp(c)\sum_j \exp(a_j)} \\
&= \frac{\exp(a_k)}{\sum_j \exp(a_j)}
\end{aligned}$$

b. With $c = -\max_j[a_j]$ we see that $\exp(a_k + c) \leq 1$. This is useful because it limits the range of the exponentiated terms and prevents overflow. For instance, without an offset, $\exp(a_k)$ where $a_k = 90$ is already too large to be represented as a 32–bit floating point number.

c. After the offset, the largest value of $\exp(a_j + c)$ is always 1. Relative to that, $\exp(-10) < \exp(-9) \approx \frac{1}{20^3} = \frac{1}{8000}$. Outputs smaller than -10 will therefore have negligible effect.

d. We take the derivative of the error for one input pattern with respect to w_{34},

$$\begin{aligned}
E(w_{34}) &= -\sum_{k=1}^{4} t_k \ln \hat{y}_k \\
\frac{\partial E}{\partial w_{34}} &= \sum_{k=1}^{4} \left[\frac{-t_k}{\hat{y}_k} \hat{y}'_k(w_{34}) \right]
\end{aligned}$$

For $\hat{y}'_k(w_{34})$, we need to differentiate the softmax for both $k = 3$ and $k \neq 3$. First, for $k = 3$, we have

$$\begin{aligned}
\hat{y}_3 &= \frac{\exp(a_3)}{\sum_k \exp(a_k)} \\
&= \frac{\exp(\sum_j w_{3j}^{(n)} x_j^{(n-1)})}{\exp(\sum_j w_{3j}^{(n)} x_j^{(n-1)}) + \sum_{k \neq 3} \exp(a_k)} \\
\hat{y}'_3(w_{34}) &= \frac{x_4^{(n-1)} \exp(a_3)(\sum_k \exp(a_k)) - x_4^{(n-1)} \exp(a_3)^2}{(\sum_k \exp(a_k))^2} \\
&= \frac{x_4^{(n-1)} \exp(a_3) \Big[\sum_k \exp(a_k) - \exp(a_3) \Big]}{(\sum_k \exp(a_k))^2} \\
&= \frac{x_4^{(n-1)} \hat{y}_3 \Big[\sum_k \exp(a_k) - \exp(a_3) \Big]}{\sum_k \exp(a_k)} \\
&= x_4^{(n-1)} \hat{y}_3 (1 - \hat{y}_3).
\end{aligned}$$

For $k \neq 3$, we have

$$
\begin{aligned}
\hat{y}_k &= \frac{\exp(a_k)}{\sum_i \exp(a_i)} \\
\hat{y}_k'(w_{34}) &= \frac{0 - x_4^{(n-1)} \exp(a_k)\exp(a_3)}{(\sum_k \exp(a_k))^2} \\
\hat{y}_k'(w_{34}) &= -x_4^{(n-1)} \hat{y}_3 \hat{y}_k
\end{aligned}
$$

so the weight update formula is

$$
\begin{aligned}
\Delta w_{34} &= -\eta \frac{\partial E}{\partial w_{34}} \\
&= -\eta \sum_{k \neq 3} \left[\frac{-t_k}{\hat{y}_k} \hat{y}_k'(w_{34}) \right] - \eta \left[\frac{-t_3}{\hat{y}_3} \hat{y}_3'(w_{34}) \right] \\
&= -\eta \sum_{k \neq 3} \left[\frac{t_k}{\hat{y}_k} x_4^{(n-1)} \hat{y}_3 \hat{y}_k \right] - \eta \left[\frac{-t_3}{\hat{y}_3} x_4^{(n-1)} \hat{y}_3 (1 - \hat{y}_3) \right] \\
&= -\eta \cdot x_4^{(n-1)} \left[\hat{y}_3 \sum_{k \neq 3} t_k + \Big[t_3(\hat{y}_3 - 1) \Big] \right] \\
&= \eta \cdot x_4^{(n-1)} \cdot [t_3 - \hat{y}_3]
\end{aligned}
$$

where the last line comes from the fact that $\sum_k t_k = 1$.

e. From the previous answer, and taking the derivative with respect to a, we get

$$
|\delta_k| = |t_k - \hat{y}_k|
$$

Since $a_4 - c = 14 - 100 = -86 < -10$, we can safely assume $\hat{y}_4^{12} \approx 0$, and therefore, $|\delta_4^{12}| \approx 0$. We then note that $|\delta_2| - |\delta_3| = \hat{y}_2 - \hat{y}_3 \propto \exp(a_2) - \exp(a_3)$, which is positive, so $|\delta_2| > |\delta_3|$. Finally, we can check $\text{sign}\,\big[|\delta_1| - |\delta_2|\big]$ to see which unit has the largest error:

$$
\begin{aligned}
\text{sign}\,\big[|\delta_1| - |\delta_2|\big] &= \text{sign}\,\Big[|1 - \hat{y}_1| - |-\hat{y}_2|\Big] \\
&= \text{sign}\left[\left| 1 - \frac{1}{\sum_k \exp(a_k - c)} \right| - \left| \frac{\exp(-3)}{\sum_k \exp(a_k - c)} \right| \right] \\
&= \text{sign}\left[\Big| \sum_k \exp(a_k - c) - 1 \Big| - \Big| \exp(-3) \Big| \right] \\
&= \text{sign}\,\Big[\big| \exp(-8) + \exp(-86) \big| \Big] \\
&= +1,
\end{aligned}
$$

so $|\delta_1|$ is the largest.

f.

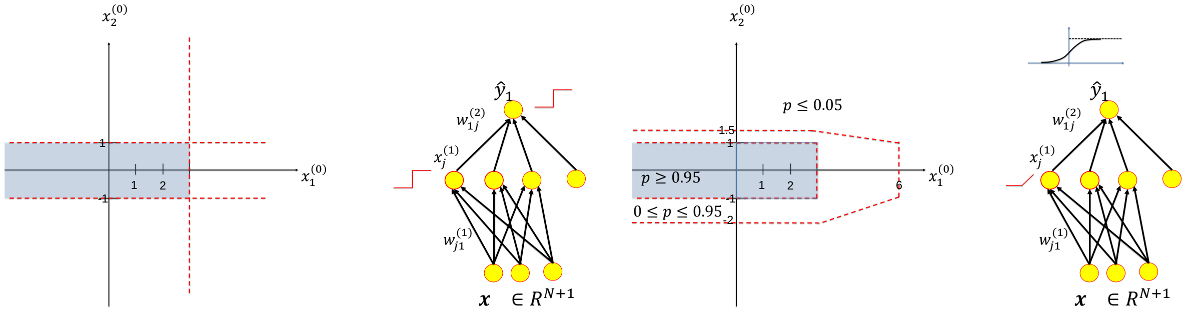
$$\begin{aligned}E(\boldsymbol{w}) &= -\sum_{\mu}\sum_{k=1}^{4} t_k^{\mu} \ln \hat{y}_k^{\mu} \\ &= -\sum_{\mu}\left[t_1^{\mu}\ln\hat{y}_1^{\mu} + t_2^{\mu}\ln\hat{y}_2^{\mu}\right] \\ &= -\sum_{\mu}\left[t_1^{\mu}\ln\hat{y}_1^{\mu} + (1-t_1^{\mu})\ln(1-\hat{y}_1^{\mu})\right]\end{aligned}$$

From the softmax definition, we have

$$\begin{aligned}\hat{y}_1 &= \frac{\exp(a_1)}{\sum_j \exp(a_j)} \\ &= \frac{\exp(a_1)}{\exp(a_1)+\exp(a_2)} \\ &= \frac{\exp(a_1-a_2)}{\exp(a_1-a_2)+1} \\ &= \frac{\exp(a)}{\exp(a)+1}\end{aligned}$$

which is the sigmoid formula with drive $a = a_1 - a_2$. Since $\hat{y}_2 = 1 - \hat{y}_1$ is fully determined by the value of $\hat{y}_1$, we only need one (sigmoidal) output.

Exercise 4. Probability modeling with rectified linear units and threshold units



A

B

Please have a look at Figs A and B. Input data $\boldsymbol{x}$ is two-dimensional. We consider a network with three hidden units and one output unit.

a. We start with the network in Fig A. All hidden units and the output unit are strict threshold functions $x^{(n)} = \theta(a)$, where θ is the Heaviside function.

All positive data samples $\boldsymbol{x}^{\mu}$ are uniformly distributed within $x_1 < 3$ and $-1 < x_2 < 1$. All negative examples our outside these bounds. Similar to the construction that was done in class for the XOR problem, your task is to construct by hand a solution such that the output is +1 for all positive examples and zero for all negative examples.

Show that there are 48 different, but equivalent solutions for the weight vectors of the hidden units, if you assume three normalized weight vectors for the hidden units; and infinitely many if you drop the normalization condition. Pick one of your solutions and put the weight values for all weights (both layers) on the graphic in A.

Hint: the dashed lines show the three linear separations that you will need for your three hidden units.

b. In Fig B, all hidden units are rectified linear units. The output unit is the sigmoidal function $g(a) = 1/[1 + \exp(-a)]$. For the sake of numerical simplicity, you should use the following approximation: $g(-3) = 0.05$ and $g(+3) = 0.95$.

The data is two-dimensional.

(i) Within the bounds $x_1 < 3$ and $-1 < x_2 < 1$ the probability $P(C_1|\boldsymbol{x})$ that an example $\boldsymbol{x}$ is indeed a positive data samples is equal or above 95 percent (and exactly 95 percent on the border). And in the area within these bounds the probabability $p(\bar{C}_1|\boldsymbol{x})$ of encountering a negative example is smaller than 5 percent (see Fig B).

The probability $p(\bar{C}_1|\boldsymbol{x})$ to find a negative example is 0.95 for

(ii) any input example on the line $x_1 \leq 3$ and $x_2 = 1.5$;

(iii) any input example on the line $x_1 \leq 3$ and $x_2 = -2.0$;

(iv) any input example on the line $x_1 = 6$ and $-1 < x_2 < +1$.

Construct a solution with piecewise linear units in the hidden layer and sigmoidal in the output layer, such that your network matches to probabilities summarized under (i) - (iv). Write the weight values for your solution directly into the graph.

Hint: There are many solutions. Pick one of these. Take one of the solutions where weight values have simple numbers. You may find it is easier to assume that in the shaded area the probability is uniformly at 0.95. Always assume that $g(-3) = 0.05$ and $g(+3) = 0.95$.

c. Optional small programming check:
Generate 10 000 positive and negative training data points according to the above rules. Take your calculated weight values as an initial condition for the weights and run a gradient descent optimizer to see whether it finds even better values.

Solution:

a. We want each of the hidden layer units to test for one of the three boundaries of the positive data region. For instance, with $\boldsymbol{w}_1^{(1)} = (-1, 0, 3)$, the unit $x_1^{(1)}$ will be activated only if $x_1 < 3$. Likewise, $\boldsymbol{w}_2^{(1)} = (0, 1, -1)$ and $\boldsymbol{w}_3^{(1)} = (0, -1, -1)$ will test that $-1 < x_2 < 1$. Finally, for the output layer, we test that all three units are activated using $\boldsymbol{w}_1^{(2)} = (1, 1, 1, 2.5)$.

Rescaling the weight vector by an arbitrary positive constant won't change the output of the Heaviside function. Therefore there are infinitely many solutions if we rescale the hidden weight vectors on the solution we have.

If the hidden weight vectors are normalized, we can't arbitrarily rescale them. In this case, there are 6 (3 permute 3) different ways to re–arrange which hidden unit is activated for which boundary condition.

As well, each hidden unit can be turned either on or off at the boundary by multiplying the weight vector by -1; if it is flipped to give an output of 0 in the positive data region, the corresponding output weight can be set to -1 and the output threshold adjusted accordingly. For instance, if we flip the input weights on unit $x_1^{(1)}$, the new output weights could be $\boldsymbol{w}_1^{(2)} = (-1, 1, 1, 1.5)$.

Since there are $2^3 = 8$ ways to set the on/off activations for the units, there are $6 \cdot 8 = 48$ possible ways to solve for the hidden layer weights (if normalized).

b. Working backwards, we want $a_{\text{out}} = -3$ at the $P(C_1|\boldsymbol{x}) = 0.05$ boundaries and $a_{\text{out}} = 3$ at the $P(C_1|\boldsymbol{x}) = 0.95$ boundaries. As before, we should have one hidden unit testing each boundary. One possibility is an output weight vector of $\boldsymbol{w}_1^{(2)} = (-6, -6, -6, -3)$. In this case, we get $a_{\text{out}} = 3$ if all of the hidden units are off, and $a_{\text{out}} \leq -3$ if any of the hidden units have activity $x_j^{(1)} \geq 1$.

For the first boundary, we want $x_1^{(1)} = 0$ at $x_1 = 3$ and $x_1^{(1)} = 1$ at $x_1 = 6$. This gives $3w_1 - w_3 = 0$ and $6w_1 - w_3 = 1$, resulting in the weight vector $\boldsymbol{w}_1^{(1)} = (1/3, 0, 1)$.

For the second boundary, we want $x_2^{(1)} = 0$ at $x_2 = 1$ and $x_2^{(1)} = 1$ at $x_2 = 1.5$. This gives $w_2 - w_3 = 0$ and $1.5w_2 - w_3 = 1$, resulting in the weight vector $\boldsymbol{w}_2^{(1)} = (0, 2, 2)$.

For the final boundary, we want $x_3^{(1)} = 0$ at $x_2 = -1$ and $x_3^{(1)} = 1$ at $x_2 = -2$. This gives $-w_2 - w_3 = 0$ and $-2w_2 - w_3 = 1$, resulting in the weight vector $\boldsymbol{w}_3^{(1)} = (0, -1, 1)$.