Machine Learning Worksheet 09

Deep Learning

1 Activation functions

Problem 1: Why do we use basis functions in neural networks? What purpose do they serve?

Problem 2: Consider a neural network with hidden-unit nonlinear sigmoid activation functions. Show that there exists an equivalent network, which computes exactly the same function, but with hidden unit activation functions given by $\tanh(x)$.

Problem 3: We already know that the derivative of the sigmoid activation function can be expressed in terms of the function value itself. Show that the derivative of the tanh activation function can also be expressed in terms of the function value itself. Why is this a useful property?

Problem 4: The error function for binary classification problems,

$$E(\boldsymbol{w}) = -\sum_{p=1}^{n}\Big\{y_p \log f(\boldsymbol{x}_p, \boldsymbol{w}) + (1-y_p)\log\big[1 - f(\boldsymbol{x}_p, \boldsymbol{w})\big]\Big\}$$

is derived for a network having a logistic sigmoid activation function so that $0 \leq f(\boldsymbol{x}_p, \boldsymbol{w}) \leq 1$, and data have target values $y \in \{0, 1\}$. Derive the corresponding error function if we consider a network having an output $-1 \leq f(\boldsymbol{x}_p, \boldsymbol{w}) \leq 1$ and target values $y \in \{-1, 1\}$. What would be the appropriate choice of activation function in this case?

2 Optimization

Problem 5: A simple neural network has as loss function

$$E(\boldsymbol{w}) := \frac{1}{m}\sum_{i=1}^{m} f(y_i - \boldsymbol{w}\cdot\boldsymbol{x}_i) + \lambda\|\boldsymbol{w}\|^2/2$$

where

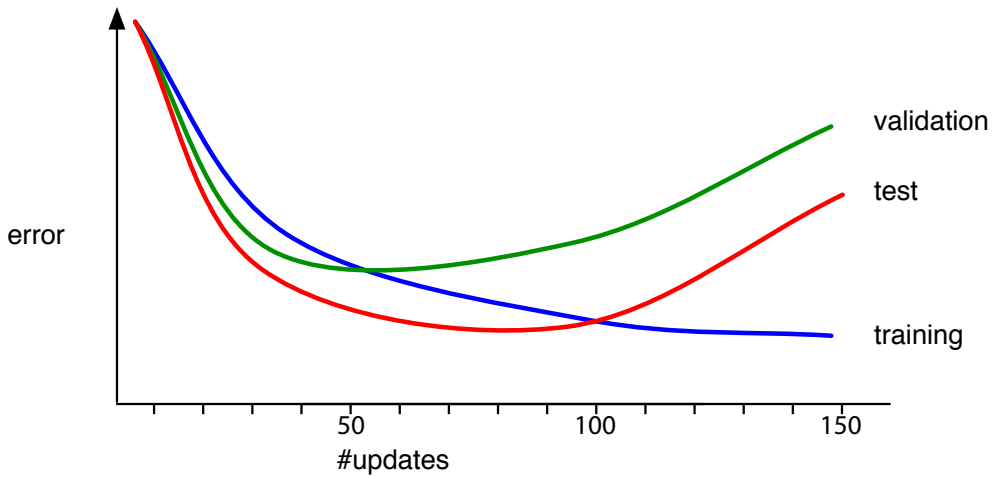
$$f(\eta) = \begin{cases} \frac{1}{2}\eta^2 & \text{if } |\eta| < 1 \\ |\eta| - \frac{1}{2} & \text{otherwise.} \end{cases}$$

Here, $\boldsymbol{x}_i \in \mathbb{R}^d$ are the data, $\boldsymbol{w} \in \mathbb{R}^d$ are the weights, and $y_i \in \mathbb{R}$ the target outputs for data points $i = 1, 2, \ldots, m$. The λ is a constant.

Compute the gradient of $E(\boldsymbol{w})$ w.r.t. $\boldsymbol{w}$, when optimising over all data.

Upload a single PDF file with your solution to Moodle by 08.01.2018, 9:59am CET. The solution must be typeset (using Latex, Word, etc.). Handwritten solutions are not accepted.

Problem 6: When do you "stop training" (give the approximate number of the update/iteration) and use the corresponding neural network weights? Explain your answer. Relate your answer to the figure below in which the data is separated in a training set, a validation set, and a test set.



3 Numerical stability

Problem 7: In machine learning you quite often come across problems which contain the following quantity

$$y = \log \sum_{i=1}^{N} e^{x_i}$$

For example if we want to calculate the log-likelihood of neural network with a softmax output we get this quantity due to the normalization constant. If you try to calculate it naively, you will quickly encounter underflows or overflows, depending on the scale of x_i. Despite working in log-space, the limited precision of computers is not enough and the result will be ∞ or $-\infty$.

To combat this issue we typically use the following identity:

$$y = \log \sum_{i=1}^{N} e^{x_i} = a + \log \sum_{i=1}^{N} e^{x_i - a}$$

for an arbitrary a. This means, you can shift the center of the exponential sum. A typical value is setting a to the maximum ($a = max_i x_i$), which forces the greatest value to be zero and even if the other values would underflow, you get a reasonable result.

Your task is to show that the identity holds.

Problem 8: Similar to the previous exercise we can compute the output of the softmax function $\pi_i = \frac{e^{x_i}}{\sum_{i=1}^{N} e^{x_i}}$ in a numerically stable way by introducing an arbitrary a:

$$\frac{e^{x_i}}{\sum_{i=1}^{N} e^{x_i}} = \frac{e^{x_i - a}}{\sum_{i=1}^{N} e^{x_i - a}}$$

often chosen $a = \max_i x_i$. Show that the above identity holds.

Problem 9: Let the logits (before applying sigmoid) of a single training example be x and the corresponding label be y. The sigmoid logistic loss (i.e. binary cross-entropy) for this example is then:

$$-(y \log(\sigma(x)) + (1 - y) \log(1 - \sigma(x)))$$

To ensure stability and avoid overflow, usually implementations use this equivalent formulation:

$$\max(x, 0) - x \cdot y + \log(1 + e^{-\text{abs}(x)})$$

You task is to show that the equivalence holds.