UNIVERSITÄT HEIDELBERG
Zukunft. Seit 1386.
IWR Interdisciplinary Center for Scientific Computing

Deep Vision
Summer Semester 20
Prof. Björn Ommer
ommer@uni-heidelberg.de

Exercise sheet 3: Sigmoid, Softmax and Gradient Descent - Revision 1

Due on *22.05.2020*, 11am.

Sandro Braun (sandro.braun@iwr.uni-heidelberg.de).

Task 1: From Sigmoid to Softmax (5P)

In this exercise, we will think deeply about the softmax and sigmoid functions. The sigmoid and softmax functions are given by:

$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)} \tag{1}$$

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{N} \exp(x_j)}, \quad i = 1, \ldots, N \tag{2}$$

- You want to build a classifier that classifies images of cats and dogs. Given images x, your model will predict $p(y|x)$, where y is $\{\text{dogs}, \text{cats}\}$. You start with a linear model $p(y|x) = \sigma(W^T x)$, where σ is an activation function and either the sigmoid or softmax function. Which activation function(s) can you use?

- Your roommate comes in and says that it actually does not matter which activation function you use for the binary problem. Proof him right (or wrong).

  Hint: start with the softmax classifier and assume that

  $$\begin{pmatrix} z_0 \\ z_1 \end{pmatrix} = \begin{pmatrix} w_0^T \\ w_1^T \end{pmatrix} x + \begin{pmatrix} b_0 \\ b_1 \end{pmatrix}, \quad p(y_i|x) = \text{softmax}(z_i). \tag{3}$$

  Then continue with the sigmoid classifier

  $$z' = w'^T x + b', \quad p(y_0|x) = \text{sigmoid}(z'). \tag{4}$$

  The classifiers are equivalent if their probabilities match. When is this the case in terms of w_i, w', b_i, b'?

- Image that you also want to include frogs in your classifier output, so that you now have $y \in \{\text{dogs}, \text{cats}, \text{frogs}\}$. Which activation function can you use now and why?

- Derive $\frac{\partial}{\partial x}\text{sigmoid}(x)$ and $\frac{\partial}{\partial x_i}\text{softmax}(x_i)$.

- Make a plot with two subplots. In subplot 1, you plot the sigmoid function, in subplot 2 you plot its derivative. Adjust the x and y range to $[-3, 3]$. Within subplot 1, plot the following function with a different color.

$$a(x) = \max(0, x). \tag{5}$$

  Within subplot 2, plot $\frac{\partial}{\partial x} a(x)$, i.e. the derivative.

  Which function, sigmoid or $a(x)$, is a better choice for training neural networks and why?

  Hint: In neural networks, we optimize an arbitrary function f by gradient descent. This requires calculating the gradient, and thus the derivative. If the gradient is zero over a large interval, the gradient *vanishes* (also known as the "vanishing gradient problem"). Vanishing gradients are bad for neural network training.

Deep Vision
Summer Semester 20
Prof. Björn Ommer
ommer@uni-heidelberg.de

Task 2: Gradient Descent **(5P)**

In this exercise we will use gradient descent to find the minimum of the Rosenbrock function. The Rosenbrock function is given by

$$f(x, y) = 100(y - x^2)^2 + (1 - x)^2 \tag{6}$$

and has a global minimizer at $(1, 1)$.

- Derive the gradient ∇f analytically.
- Write a small script that evaluates f an ∇f. Then perform 30 gradient descent steps from the initial point $(0.5, 2)$ with a fixed step size. Evaluate 10 step sizes in a range between $1.0 \cdot 10^{-2}$ and $1.0 \cdot 10^{-5}$ in logarithmic scale and compare their convergence rate in terms of $\|.\|^2$ distance from the global minimizer. Reflect on how easy it was to choose an appropriate step size.

  Hint: The simplest gradient descent update rule is given by

  $$x_{t+1} = x_t - \eta(\nabla f)(x_t), \tag{7}$$

  where t is the iteration time-step and η is a fixed step size (also called learning rate).
- Build an optimizer that automatically adjusts the step size. Then rerun the same experiments with the same range of step sizes. Reflect again on how easy it was to choose an appropriate step size.

  Hint: A popular way to do this is to use Adaptive Moment Estimation (ADAM) [1].

  $$m_t = \beta_1 m_{t-1} + (1 - \beta_1)(\nabla f)_t \tag{8}$$
  $$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla f)_t^2 \tag{9}$$
  $$x_{t+1} = x_t - \frac{\eta}{\sqrt{v_t} + \epsilon} m_t \tag{10}$$
  $$\beta_1 = 0.9 \tag{11}$$
  $$\beta_2 = 0.999 \tag{12}$$
  $$m_0 = 0 \tag{13}$$
  $$v_0 = 0 \tag{14}$$
  $$\epsilon = 1.0 \cdot 10^{-6} \tag{15}$$

  All operations are elementwise, i.e. $(A)^2$ means elementwise square of the matrix A. Basically, a running mean of the first and second order statistics are kept at each iteration step t.

- Think what happens in the very first steps of the ADAM optimization scheme that might cause problems.

  Hint: Start with $t = 0$ and write down the value of m_0, v_0. Perform one update step : $t \leftarrow t + 1$. Write down the value of m_1, v_1 and compare with the previous step.

- The authors of [1] found a way to solve the problem you just explained, by modifying the update rule.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)(\nabla f)_t \tag{16}$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla f)_t^2 \tag{17}$$
$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{18}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{19}$$
$$x_{t+1} = x_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t. \tag{20}$$

  Explain how this solves the problem at timestep $t = 1$.

  Hint: Start with $t = 0$ and write down the value of m_0, v_0. Perform one update step : $t \leftarrow t + 1$. Write down the value of m_1, v_1, $\hat{m}_1$, $\hat{v}_1$. What has changed?

Note: Submit exactly one ZIP file and one PDF file via Moodle before the deadline. The ZIP file should contain your executable code. Make sure that it runs on different operating systems and use relative paths. Non-trivial sections of your code should be explained with short comments, and variables should have self-explanatory names. The PDF file should contain your written code, all figures, explanations and answers to questions. Make sure that plots have informative axis labels, legends and captions.
Changelog:

- 2020.05.15 - In Problem 2, the learning rates should be $1.0 \cdot 10^{-2}$ and $1.0 \cdot 10^{-5}$ and not $1.0 \cdot 10^2$ and $1.0 \cdot 10^5$.
- 2020.05.15 - In Problem 1, the softmax function is given by $\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{N} \exp(x_j)}, \quad i = 1, \ldots, N$. The index in the sum is j not i.

References

[1] D. P. Kingma and J. L. Ba, *Adam: a Method for Stochastic Optimization.* , International Conference on Learning Representations 2015