

Assignment 1

Due date: Feb 20 (11.58 PM CT)

Turn in solutions as pdf(s) and ipynb (Python 2 preferred) file(s) on Blackboard.

Note: Answer the following questions concisely, in complete sentences and with full clarity. If in doubt, ask classmates using Slack. Across group collaboration is not allowed. Cite all your sources (see the Syllabus for these and other pointers).

1 Backpropagation (5pt)

1. Draw the computation graph for the following function: $f(a, b, c, d, e) = \frac{1}{(1+(a^b+c^d)*e)^2}$. Compute the gradient of the function with respect to its inputs at $(a, b, c, d, e) = (1, 1, 1, 1, 1)$.

2 Gradient Descent (15pt)

1. Write a function to compute the mean squared error between a prediction and ground truth assuming both are *numpy* arrays (see python module *numpy*).
2. Consider a model: $y = mx + c$, where the model parameter $m = 1$ and parameter $c = 0$ and $x \in (0, 1)$. Plot the function using matplotlib.
3. Generate example data by drawing $N = 100$ uniform values from the range in which x lies and compute the corresponding y to get $\{x_i, y_i\}_{i=1}^N$.
4. Assuming that you do not know the model **parameters**, use backpropagation and gradient descent updates to find the model parameters (choose an appropriate learning rate). The loss function will be the mean squared error.
5. Plot the error in the estimates as a function of the number of iterations of gradient update. Change the learning rate and plot another curve on the previous plot.
6. Do steps 3-5 when the model is $y = m_1x + m_2x^2 + c$ and the true parameters are $m_1 = 0.5$, $m_2 = 1$ and $c = 1$. And $x \in (0, 1)$. Also, plot the ground truth function. Compare and contrast the plot with the previous one.
7. Do steps 3-5 when the model is $y = \tanh(m * x + c)$ and the true parameters are $m = 1$ and $c = 2$. And $x \in (0, 2)$. Also, plot the ground truth function.

3 ML Basics (5pt)

1. Write a function to compute the (**multiclass**) logistic loss (**also called the cross-entropy loss**) given the parameters (W, b) of a linear model (as *numpy* arrays) and an example (x, y) .
2. Add an ℓ_1 regularization and an ℓ_2 regularization to the loss function.

4 Classification pipeline (20pt)

1. Generate data from *assignment1_code.py*.
2. Split the data into test and train (20%:80%).
3. Build a linear classifier assuming the multiclass logistic loss and an ℓ_2 regularization for the weights only. Report the prediction accuracy on the training data and the test data and show appropriate plots.
4. Introduce a cross validation scheme and justify your choice of parameters. What is the validation accuracy compare to the test accuracy.
5. What is the sensitivity of the model's performance to different learning rates and the number of gradient descent iterations. Describe via suitable plots.
6. What is the sensitivity of the model's performance to different regularization parameter values. Find the best regularization parameter using an exhaustive search procedure. Describe your choice via suitable plots. What is the performance difference between using regularization and no regularization?
7. What is the sensitivity of the model's performance with respect to a different test train split (e.g., 50%:50%).

5 Feedforward Neural Network (25pt)

1. Repeat Question 4 when the model is a 2-layer feedforward neural network (i.e., 1 hidden layer with $f(x, W_1, b_1, W_2, b_2) = W_2 \max(0, W_1 x + b_1) + b_2$). Use hand-computed backpropagation equations.
2. Repeat item 1 when the non-linearity is changed to leaky ReLU ($f(x) = x$ if $x > 0$, else $f(x) = 0.01 * x$).
3. Repeat Question 4 when the model is $f(x, W_1, b_1, W_2, b_2, W_3, b_3) = W_3 \max(W_2 x + b_2, W_1 x + b_1) + b_3$ (this is called the maxout non-linearity).
4. In items 1-3, what happens when the number of hidden units chosen is much smaller. Similarly, what happens when the number of hidden units chosen is much higher?

6 Real datasets and Keras (20pt)

1. Repeat Question 5.1 with the MNIST dataset. Describe the dataset properties as well. What did you do differently compared to the 2D synthetic dataset from *assignment1_code.py*?
2. Repeat Question 5.1 with the CIFAR-10 dataset. Describe the dataset properties as well. What did you do differently compared to the MNIST dataset?

3. Build the linear classifier and the feedforward neural networks in Questions 4 and 5 using the python module *Keras* (with backend *Tensorflow* for automatic back-propagation) and check the performances differences when the optimizer is SGD. What could be the cause of the differences, if any?

7 CNNs and finetuning (10pt)

1. Run the [VGG pretrained model](#) in Keras for sample images from the Cats&Dogs dataset and display the top 5 predictions for a random sample of 4 images (show the images and the prediction labels).
2. Use the VGG output (1000 dimensional) as the feature vector instead of the image itself and repeat question 4.3 for the Cats&Dogs dataset. Use Keras.

Datasets

- The 2D simulated data is available on Blackboard (see *assignment1_code.py*).
- MNIST image data: Original data can be found [here](#). Processed dataset can be obtained from Tensorflow directly (see [this](#) or [this](#) which hosts [mnist.pkl.gz](#)).
- CIFAR-10 dataset: Original data can be found [here](#) (see the [python version](#)). Another example way to obtain the processed dataset can be found [here](#).
- Cats&Dogs dataset: Original dataset can be obtained from [Kaggle](#) (you may have to sign up).