Theoretical Tasks

Task 0.1 Convolution

Theoretical Background

Read the following two blog posts:

- What are convolutions?
- Convolutions and Neural Networks

Practice:
Consider the following image I, represented as a matrix:

$$I = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & -3 & -4 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

And the following kernel k, represented as a matrix:

$$k = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Calculate a same convolution $I * k$ as described in Convolutions and Neural Networks above. Use zero padding for handling the margins. Since it's a same convolution, use (1, 1) stride. Do it by hand.

Task 0.2 Non Linearity

Theoretical Background

In general, Convolutional layers are followed by a nonlinearity. Common non-linear functions are sigmoid-like (sigmoid, tanh, softsign, ...) and ReLU-like (ReLU, LeakyReLU, ...).

Task 0.3 Max Pooling

Theoretical Background

Max Pooling divides the input image into sections of a given size and returns the biggest value in each section. Apply valid Max Pooling with a filter size of (2, 2) on the result of the previous task.

Task 0.4 Flattening

Theoretical Background

Flattening reshapes a matrix into a one-dimensional vector by putting all the rows of the image in one line. Practice: Flatten the result of the previous task.

Task 0.5 Fully Connected Layer

Theoretical Background

After extracting features, a fully connected layer is used for classification. Practice: Perform a matrix multiplication for a fully connected layer.

Task 0.6 SoftMax

Theoretical Background

Softmax operation transforms the raw output of the network into probabilities. The highest number after softmax is selected as the output class. Practice: Apply softmax to the output of the previous task and determine the output class.

Task 0.7 Loss Functions

The following formulas are useful for doing the exercise, where n denotes the length of both the prediction vector y and the ground truth vector g.

Cross-Entropy Loss (or Logistic Loss) $H(y, g) = -\sum_i g_i \log(y_i)$

Mean Squared Error-Loss $MSE(y, g) = \frac{1}{n}\sum_i (y_i - g_i)^2$

Hinge Loss (or SVM Loss) $SVM(y, j) = \sum_{i|i \neq j} \max(0, y_i - y_j + 1)$

Task: Consider the following two vectors: $g = [0, 1, 0]$ $y = [0.25, 0.6, 0.15]$
Calculate the values:

- Cross-Entropy Loss
- Mean Squared Error Loss
- Hinge Loss

Resources:

- What's an intuitive way to think of cross entropy?

- Section 3.13 from the Deep Learning Book
- Notes from CS231n

Evaluation Metrics

Task 0.1 Theoretical Foundations

Typically people refer to accuracy as THE evaluation metric, but there are a lot of evaluation metrics which can be better suited than accuracy depending on the task/dataset.

1. In which situation using accuracy is not necessarily a good idea?
2. What part of the formula for computing the accuracy makes it less desirable than the Jaccard Index (Intersection Over Union) in a multi-class setting?
3. What is the difference between Jaccard Index (Intersection Over Union) and F1-Measure? Which one is more suited to measure performance of NNs?

Task 0.2 Practice

In this part of the exercise we want to compute some common alternatives which can be used instead of accuracy. We'll take an example from a real case scenario of layout analysis at pixels level of historical documents.

Given the following prediction and ground truth (note: this is a multi-class and multi-label scenario!), where B stays for background, T for text, D for decoration and C for comment.

	1	2	3	4	5	6	7	8
GT	B	T	B	B	TD	TD	TD	TD
P	B	B	TD	BD	BC	TC	T	TD

Compute the class frequencies and the following metrics per class:

- Jaccard Index
- Precision
- Recall
- F1-measure

Then compute their mean in two different ways: once with class balance (sum of per class values divided by number of classes) and once with the class frequencies

Resources:

- Jaccard Index (or Intersection over Union): https://en.wikipedia.org/wiki/Jaccard_index
- Exact Match (and others metrics): https://en.wikipedia.org/wiki/Multi-label_classification
- Precision and Recall: https://en.wikipedia.org/wiki/Precision_and_recall